재사용성 향상을 위한 임베디드 소프트웨어의 동적 가변성 설계 기법

(A Dynamic Variability Design Technique of Embedded Software for Improving Reusability)

김 철 진 † 조 은 숙 ††
(Chul-Jin Kim) (Eun-Sook Cho)

요 약 홈 네트워크 시스템에서는 가전 기기들이 각기 제조된 회사에 따라 서로 다른 제어 데이타 포맷을 가지고 있으며, 디지털 기기(디바이스)의 유형과 프로토콜이 다양하다. 또한 다양한 디바이스들 간의 상호 운영 환경이 상이하다. 홈 네트워크 시스템이 이와 같은 특성들로 인해 데이타 호환성나 동시 제어, 동적 플러그-인과 같은 기능들이 미약하게 지원되고 있다. 이로 인해 홈 네트워크 시스템의 재사용성은 매우 빈약한 상태이다.

본 논문에서는 홈 네트워크 시스템의 재사용성을 향상시키기 위해 가변성의 범위를 폭넓게 다룰 수 있는 재사용 프레임워크와 이를 기반으로 한 가변성 설계 기법을 제시한다. 즉, 홈 네트워크 시스템의 다양한 부분들을 가변부로 추출하여 이를 가변성 유형으로 분류 정의하고 이를 재사용 할 수 있는 프레임워크를 제안하며, 이러한 프레임워크를 기반으로 재사용성을 향상시키기 하기 위한 가변성 설계 기법을 제안한다. 재사용 프레임워크를 실제 홈네트워크 시스템 설계에 적용함으로써 다양한 도메인에 재사용될 수 있음을 증명한다.

키워드 : 홈 네트워크 시스템, 재사용성, 재사용 프레임워크, 가변성

Abstract Devices of home network system have different control data formats according to each product company. Therefore, types or protocols of digital devices are various. Also, interaction operating environments are different among various devices. These characteristics of home network system don't support sufficiently functionalities such as data comparability, concurrent control, dynamic plug-in, and so on. That is, the degree of reusability of home network system is very poor.

This paper proposes a framework which can be coverable to the scope of reusability widely and a design technique based on framework in order to improve reusability. That is, we extract various parts of home network systems as variation points, classify and define these as variation types, propose a framework which can be reusable those, and proposes a design technique of variability to improve reusability. Finally, proposed technique can be reusable to various domains by applying proposed reusability framework into real home network system's design.

Key words : Home Network System, Reusability, Reusable Framework, Variability

· 본 논문은 2007년도 서일대학 학술연구비에 의해 연구되었음

† 정 회 원 : 삼성전자 디지털 미디어 총괄 책임연구원
cjkim777@gmail.com
†† 종신회원 : 서일대학 소프트웨어과 교수
escho@seoil.ac.kr
(Corresponding author임)

논문접수 : 2008년 6월 11일
심사완료 : 2008년 12월 12일

1. 서 론

임베디드 소프트웨어는 마이크로프로세서 위에 내장되어 산업 및 군사용 제어기기, 디지털 가전기기, 자동센서 장비 등의 기능을 다양화하고 부가가치를 높이는 핵심 소프트웨어를 의미하며, 임베디드 운영체제, 임베디드 미들웨어, 임베디드 응용 소프트웨어, 임베디드 소프트웨어 개발 도구 등 다양한 영역들이 여기에 해당된다고 볼 수 있다. 즉, 임베디드 소프트웨어는 우리가 일상에 쉽게 접하는 휴대폰, TV, 세탁기, 기차, 비행기, 엘리베이터 등의 제품 안에 내장된 임베디드 시스템에

서 하드웨어를 제외한 나머지 부분이라고 말할 수 있다[1-3]. 이러한 임베디드 시스템 가운데 현재 대표적으로 많이 개발되어 있는 형태가 홈 네트워크 시스템이라 할 수 있다. 홈 네트워크 시스템이란 가정 내의 가전, PC, 통신 기기 등의 기기 간의 네트워크 연결을 통해서 서로 데이타를 주고받을 수 있는 시스템을 말한다. 즉, 서로 다른 가전기기를 일괄적으로 관리하고 이를 제어하는 시스템을 의미한다[4-6].

임베디드 소프트웨어 개발에 있어서 소프트웨어 공학적인 기법들이 미흡하게 적용되고 있으며, 재사용을 위한 체계적인 공정이나 기법 연구가 미흡한 상태이다. 특히, 홈 네트워크 시스템의 경우는 댁내에 존재하는 다양한 디바이스 장치나 프로토콜 등에 대한 가변성이 반드시 고려되어 설계 되어야, 임베디드 소프트웨어의 재사용성이나 개발 생산성이 향상될 수 있다. 그러나 현재 개발되어 있는 홈 네트워크 시스템에서는 이러한 부분이 미흡한 상태로 설계 및 개발되고 있다[5-10]. 본 논문에서는 임베디드 소프트웨어의 재사용 측면에서 가변성 부분의 재사용성을 향상시킬 수 있는 재사용 프레임워크를 제안하며, 이를 기반으로 하는 홈 네트워크 임베디드 시스템 설계 기법을 제안한다.

이러한 재사용 프레임워크를 기반으로 임베디드 소프트웨어를 개발할 경우, 홈 네트워크 소프트웨어의 특성 가운데 하나인 다양한 디바이스들에 대한 행위 가변성을 동적으로 지원할 수 있을 것이다. 이는 결과적으로 임베디드 시스템의 재사용성을 향상시키는 결과를 가져올 뿐만 아니라 시스템의 복잡도 또한 줄일 수 있는 효과를 기대할 수 있을 것이다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구로 현재 홈 네트워크 시스템 개발에 있어서의 한계점과 기존 가변성 설계 기법을 홈 네트워크 시스템에 적용하는데 있어서의 한계점을 살펴본다. 3장에서는 본 논문에서 제시하는 홈 네트워크 시스템 가변부의 분류와 이를 기반으로 한 재사용 프레임워크 및 기법을 제시한다. 4장에서는 본 논문에서 제시하는 재사용 프레임워크를 기반으로 재사용 가능한 홈 네트워크 임베디드 시스템 설계 사례와 기존의 객체지향 설계 사례를 비교하여 재사용성을 증명한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

2.1 기존 홈 네트워크 시스템 개발의 한계성

기존의 홈 네트워크 시스템에서는 각각의 기기들이 네트워크 설정을 통해 서버에 연결되는 클라이언트/서버 모델을 채택하였다[4,5]. 이 방식은 네트워크 변경 시 각 기기마다 네트워크 설정을 바꿔주어야 하므로 이동성을 지원하지 못한다. 그리고 IPv4 주소 체계를 이용한 홈 네트워크에서는 홈 네트워크에 새로 가입된 기기에 대한 할당이 용이하지 않다. 또한 가전기기들은 각기 제조된 회사의 선택에 따른 제어 데이타 포맷을 가지고 있다. 그리하여 기존의 홈 네트워크 시스템에서는 다른 데이타 포맷을 가진 기기들은 하나의 네트워크를 이룰 수 없다.

홈 네트워크 시스템의 가장 큰 문제점 가운데 하나는 디지털 기기(디바이스) 간의 상호 운영성 문제이다. 즉 디지털 기기간의 상호 운영성이 보장되어야 하는 것이다. 이를 해결하기 위한 방안으로 홈 네트워크 미들웨어가 개발되어 있지만 아직 실용화 단계로서 미흡한 수준에 있다. 두 번째 문제점으로는 홈 네트워크 시스템은 동시에 다양한 가전 디바이스를 제어하기 위한 메커니즘의 부재이다. 즉, 하나의 이벤트로부터 서로 다른 디바이스를 동시에 제어할 수 있는 동시 제어 메커니즘이 제공되어야 한다. 특히 이 부분은 소프트웨어적으로 처리해야 하는데, 이를 지원하기 위해서는 다양한 디바이스들에 대해 동적으로 행위를 제어할 수 있는 동적 커스터마이제이션 기법이 고려되어야 한다.

세번째 문제점으로는 홈 네트워크 시스템 제품 계열을 볼 때 특정 시스템 마다 적용되는 프로토콜이나 디바이스 유형, 서비스 형태가 공통적인 부분과 특정 제품에 특화된 부분이 존재한다. 이러한 부분을 효율적으로 지원하기 위해서는 공통된 부분은 재사용 단위 컴포넌트로 설계해서 개발해야 하고, 가변적인 부분은 동적으로 플러그-인 되어 처리될 수 있도록 설계되어야 한다. 그러나 대부분의 솔루션들은 제품별로 각각 설계되고, 개발되어 있는 형태를 취하고 있기 때문에, 동일한 디바이스나 서비스에 대한 재사용 비율이 매우 미흡한 실정이다. 본 연구에서는 이러한 부분을 향상시키기 위한 전략으로 재사용 프레임워크와 재사용성 향상 기법을 제시한다.

2.2 홈 네트워크 시스템 적용에 있어서 기존 가변성 설계의 한계점

재사용 가능한 컴포넌트는 다양한 도메인 과제의 요구사항을 대응할 수 있는 가변부(Variation Point)를 제공해야만 한다. 이러한 가변부를 통해 컴포넌트의 재사용성을 향상시킬 수 있으며 더 많은 다양한 도메인 과제에 적용될 수 있다[11-14]. 그러나 기존에 제공되고 있는 가변부를 위한 메커니즘 및 설계 패턴들은[15,16] 빠르게 변화하는 시스템의 다양한 요구사양을 충족시키는데 제한적이며 미흡하다. 예를 들면, 다양한 프로토콜 타입이나 서로 다른 유형의 디바이스 기종들에 대한 동적 연동 등과 같은 가변성들을 고려해서 설계하고 있지 않다. 이는 결국 동일 인터페이스를 구현한 클래스를 동적으로 변경할 수 없으며, 인터페이스가 다른 다양한

요구 기능을 제공할 수 없게 된다. 또한 기존 가변성 설계 기법에서는 특정 시스템에 특화된 설계기법을 반영하고 있지 않고, 전반적인 가변성 설계 기법을 제시하고 있다. 따라서 본 논문에서는 홈 네트워크 시스템에 적용할 수 있는 기법으로 제시하고자 한다.

본 논문에서는 임베디드 시스템의 다양한 요구사항에 대응할 수 있는 재사용 프레임워크를 제안한다. 본 재사용 프레임워크는 가변부로 설계된 부분에 반영할 수 있으며 동적으로 가변처리를 할 수 있도록 하여 개발 시점뿐만 아니라 운영 시점에도 다양한 요구사항에 대응 가능하도록 한다. 이러한 홈 네크워크 임베디드 시스템의 재사용 프레임워크는 홈 네트워크 시스템의 재사용성을 높일 수 있으며 근본적으로 개발 생산성을 향상시킬 수 있을 것이다.

2.3 기존 동적 커스터마이제이션 한계점

동적 커스터마이제이션 기법에 관련된 연구로서 "Using Component Composition for Self-customizable Systems"[17]와 "Component Interface Pattern"[18]가 있다. 그러나 두 논문은 모두 복합 컴포넌트를 개발할 때 커스터마이제이션을 하기 위한 기법들을 다루고 있기 때문에 컴포넌트 내부의 행위에 대한 커스터마이제이션 기법은 부분적으로 다루고 있다. [17]의 논문은 컴포넌트들 간에 예상하지 못한 변경을 하기 위해 요구되는 인터페이스에 대한 정의를 하여 동적으로 커스터마이제이션이 되도록 하기 위한 기법이다. [18]기법도 또한 컴포넌트를 조합하기 위해 컴포넌트의 인터페이스에 제공 인터페이스와 요구 인터페이스 뿐만 아니라 구조적 설명(Structural Description)과 행위적 명세(Behavior Specification)을 통해 표현한다. 구조적 설명은 컴포넌트 인터페이스의 서비스 접근 포인트인 채널(Communication Channel)를 이용하여 표현하며 행위적 명세는 패트리 넷(Petri net)을 이용하여 컴포넌트들 간의 연결과 협업을 명세한다. [17]과 [18]의 연구는 모두 복합 컴포넌트를 동적으로 조합하기 위한 명세 기법 및 패턴을 제시하고 있으며 단일 컴포넌트에 대한 가변부 커스터마이제이션 기법은 제시하지 못하고 있다.

3. 홈 네트워크 임베디드 시스템의 가변성 설계

본 논문에서 제시하는 홈 네트워크 임베디드 시스템의 재사용성을 향상시키기 위한 프레임워크는 홈 네트워크 시스템의 다양한 요구사항을 만족시켜줄 수 있을 것이다. 또한, 경량(Light Weight)의 홈 네트워크 임베디드 시스템의 개발이 가능하며 개발 시점뿐만 아니라 운영 중에 동적으로 변경할 수 있도록 한다. 본 기법은 홈 네트워크 시스템의 하드웨어 변경을 제공하기 위해 임베디드 소프트웨어 영역에서 가변성을 제공할 수 있다.

3.1 가변부 정의

공통부와 가변부는 도메인 분석을 통해 추출이 된다. 공통부은 수정 없이 재사용될 수 있는 기능이며 가변부은 도메인 마다 변경이 발생하는 기능이므로 변경될 수 있도록 제공해야 재사용 가능하다. 따라서, 가변부 설계는 특정 패턴이나 프레임워크를 이용하여 재사용될 수 있도록 설계해야 하지만, 공통부 설계는 재사용될 수 있는 제공 인터페이스(Provided Interface)만을 설계하여 제공한다.

홈 네트워크 시스템에서는 가전기기들이 각기 제조된 회사의 선택에 따른 제어 데이타 포맷을 가지고 있으며, 디지털 기기(디바이스)의 유형과 프로토콜이 다양하다. 또한 다양한 디바이스들 간의 상호 운영 환경이 상이하다. 이러한 다양한 부분들을 본 논문에서 홈 네트워크 시스템의 가변부로 추출하여 다음과 같이 가변성 유형으로 분류 정의한다. 그림 1에서와 같이 홈 네트워크 임베디드 시스템의 가변성 종류에는 제어(Control), 디바이스 드라이버(Device Driver), 디바이스 프로토콜(Device Protocol), AV 코덱(AV Codec), 운영체제(OS) 등이 있다. 여기서 제어 가변부는 동일 행위에 대해 디바이스별 처리 메커니즘의 다르게 처리하기 위한 유형이다. 디바이스 드라이버 가변부와 디바이스 프로토콜 가변부는 홈 네트워크 시스템을 구성하는 디바이스들과 이에 특화된 프로토콜을 처리하기 위해 정의한 가변부이다. AV 코덱 가변부는 기기마다 처리하는 데이타 포맷의 다양화로 인해 각 기기에 적절한 형태로 데이타 포맷을 변환해 주기 위한 가변부이다. 운영체제 가변부는 홈 네트워크 시스템에서 다양한 디바이스들 간의 상호 연동을 위한 운영 메커니즘을 처리하기 위한 가변부이다.

가변부는 특정 요구사항에 대해 변경이 가능하도록 제공하는 영역이다. 가변부는 임베디드 소프트웨어의 다른 영역에서 사용되며 가변부의 변경은 다른 영역에 전혀 영향을 주지 않도록 제공해야 한다. 가변부는 동일한 서비스(인터페이스)에 대해 서로 다른 기능(알고리즘, 프로토콜, 등)으로 제공할 수 있으며 가변부를 사용하는 영역은 동일한 서비스에 대해 다른 기능으로 변경하여 사용할 수 있다[19,20].

3.2 홈 네트워크 임베디드 시스템의 가변성 설계를 위한 재사용 프레임워크

홈 네트워크 임베디드 시스템의 가변부 처리 메커니즘은 가변부 사용 클래스들이 재사용 프레임워크를 통해 가변부를 접근한다. 홈 네트워크 시스템의 가변성에 해당하는 디바이스 프로토콜, 디바이스 드라이버, AV 코덱, OS, 제어(시그널)를 가변적으로 접근할 수 있는 메커니즘을 제공한다.

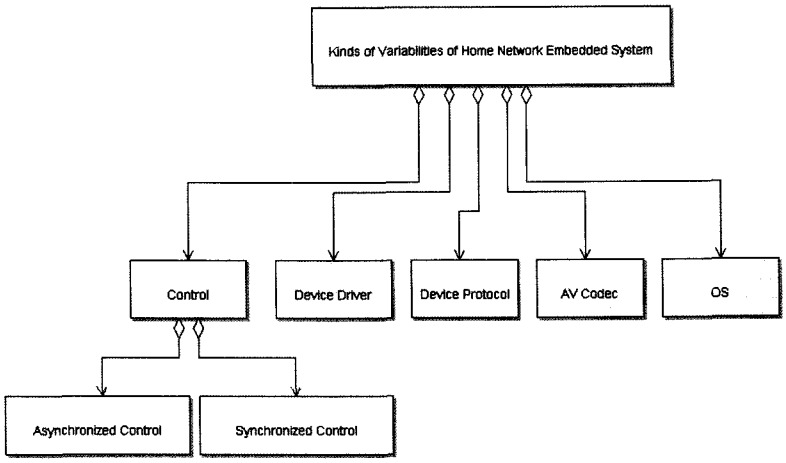

그림 1 홈 네트워크 임베디드 시스템의 가변성 종류

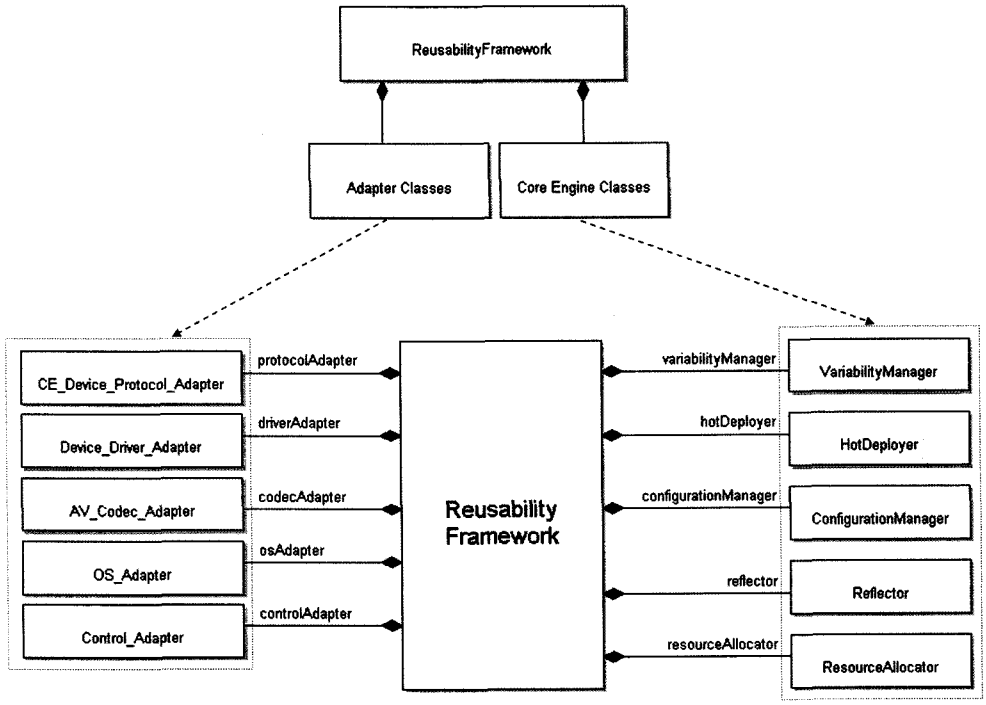

그림 2 홈 네트워크 임베디드 시스템을 위한 재사용 프레임워크 구성 요소

홈 네크워크 임베디드 시스템을 위한 재사용 프레임워크의 구성은 그림 2와 같다. 홈 네트워크 시스템의 가변부들을 중계하기 어댑터와 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. 가변부 처리를 위한 핵심 클래스는 가변성 관리기(Variability Manager), 리플렉터(Reflector), 설정 관리기(Configuration Manager), 동적 전개기(Hot Deployer), 그리고 자원 할당기(Resource Allocator)로 구성된다.

가변부의 변경을 제공하기 위해 재사용 프레임워크는 인터페이스를 제공한다. 서비스 인터페이스는 재사용 프레임워크의 요구 인터페이스를 통해 설정된 서비스를 제공한다. 홈 네트워크 시스템의 가변성에 대한 요구 인터페이스는 다양한 가변 기능으로 설정될 수 있으며, 이

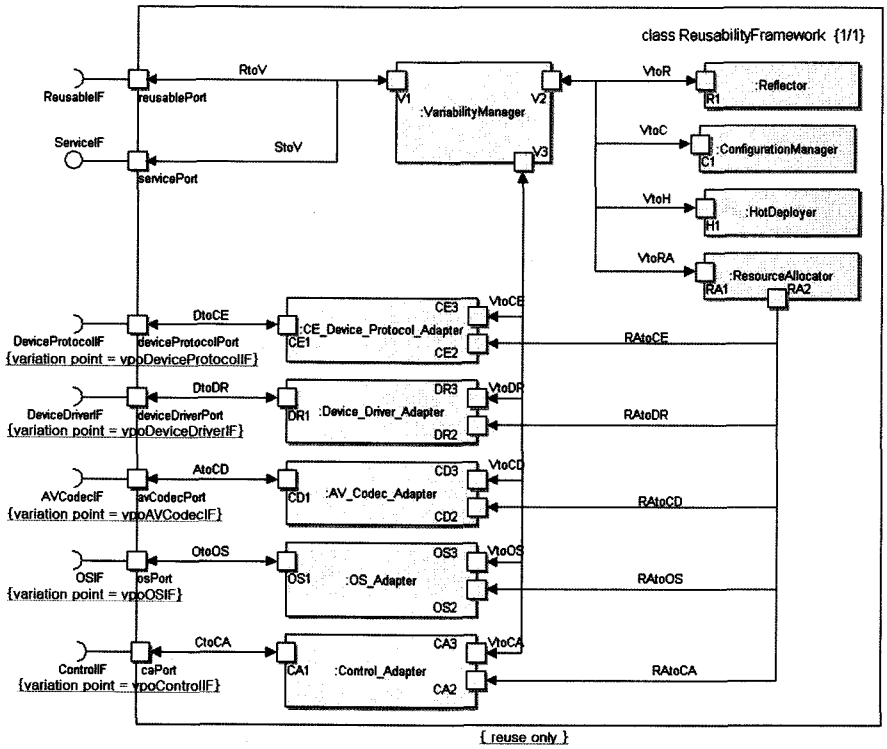

그림 3 홈 네트워크 임베디드 소프트웨어를 위한 재사용 프레임워크 내부 구조

러한 설정은 재사용 요구 인터페이스를 통해 설정된다.

홈 네트워크 시스템의 가변부에 대한 재사용 프레임워크는 어댑터와 핵심 클래스를 통해 동적으로 홈 네트워크 서비스를 제공할 수 있다. 이러한 재사용 프레임워크의 어댑터와 핵심 클래스들의 내부 구조는 그림 3과 같다. 홈 네트워크 시스템의 가변성에 대한 어댑터들은 재사용 요구 인터페이스를 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. 홈 네트워크 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.

가변성의 표기는 VPM(Variation Point Model)[21] 기법을 활용하여 그림 3에서 "{variation point = vpo-DeviceProtocolIF}"와 같은 방식으로 표기한다.

재사용 프레임워크의 내부 구성 요소들에 대한 특징은 다음과 같다.

• 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.

```
Object result = VariabilityManager.execute( _VARIABILITY_NAME_ , _PARAMETER_ );
{ variation point = vpo_VARIABILITY_NAME_ }
{ variation point = vpo_PARAMETER_ }
```

_VARIABILITY_NAME_ : 가변부 식별자

그림 4 가변부 사용 영역에서의 가변부 관리기 실행 코드

가변부 사용 영역에서는 가변부를 사용하기 위해 그림 4와 같이 가변성 관리기를 통해 가변부를 호출한다. 가변부 호출 시 가변부에 대한 식별자와 입력 데이타를 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.

• 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. 그림 4와 같이 가변부 사용 영역에서 전달된 가변부 식별자를 기반으로 설정 관리기는 가변부의 상세한 가변부 메타정보를 얻는다.

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.

• 메타정보 저장소

메타정보 저장소는 가변부의 메타정보를 포함하고 있는 저장소로서 XML 기반의 가변부 정보를 관리한다.

```
...
<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 5 가변부 메타정보

그림 5에서와 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 정의 한다. 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)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.

• 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 있으며[22,23], 본 재사용 프레임워크의 리플렉터는 이러한 메커니즘을 커스터마이즈하여 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 6에서와 같이 리플렉터는 가변성 어댑터 식별자를 이용하여 물리적인 가변성 어댑터를 실행한다. 또한 가변성 어댑터는 리플렉터를 통해 가변성 클래스를 실행한다. 가변부를 변경하고자 할 때는 가변부 메타정보만 변경하면 되며 그림 6의 가변부 사용 구조는 전혀 영향을 받지 않는다. 단지 전달되는 파라메터의 타입은 고려할 여지가 있다.

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

• 가변성 어댑터

가변성 어댑터는 가변부를 대행하는 중계자 역할을 하며, 선택된 가변성 어댑터는 메타정보에 정의된 클래스를 호출한다.

재사용 프레임워크의 가변성 어댑터는 다양한 가변처리를 지원할 수 있도록 요구 인터페이스(Required Interface)로 정의되며, 이러한 요구 인터페이스에 맞는 클래스를 통해 가변부를 제공한다. 그림 7은 홈 네트워크 시스템의 가변성인 디바이스 프로토콜, AV 코덱, 운영체제 가변성에 대한 가변부 설계 구조를 나타낸다. 홈 네트워크 시스템은 다양한 디바이스 프로토콜을 통해 다양한 디바이스를 지원하기 때문에 디바이스에 맞는 프로토콜인 RS485나 PSTN(Public Switched Telephone Network), PLC(Power Line Communication) 등을 가변적으로 선택할 수 있도록 제공해야 한다. 이러한 디바이스 프로토콜을 제공하는 클래스들은 가변성 요구 인터페이스인 'DeviceProtocolIF'를 만족하도록 설계되어야 한다. AV 코덱과 운영체제 가변성도 이와 동일하게 요구 인터페이스를 구현한 가변부를 설계한다.

어댑터는 메타정보가 변경되면 물리적인 클래스도 변

• 가변부 사용 클래스

```
Object result = VariabilityManager.execute( _VARIABILITY_NAME_ , _PARAMETER_ ) ;
{ variation point = vpo_VARIABILITY_NAME_ }
{ variation point = vpo_PARAMETER_ }
```

_VARIABILITY_NAME_ : 가변성 메타정보 식별자
PARAMETER : 전달 데이터 (예) 객체 배열

• 가변성 관리기 클래스

```
// 설정 관리기에 의해 가변부 메타정보 로딩

Object result = Reflector.delegate( _ADAPTER_NAME_ , _PARAMETER_ ) ;
{ variation point = vpo_ADAPTER_NAME_ }
{ variation point = vpo_PARAMETER_ }
Return result;
```

_ADAPTER_NAME_ : 가변성 어댑터 식별자

• 가변성 어댑터 클래스

```
// 가변부 클래스 및 행위 메타정보 로딩

Object result = Reflector.execute( _CLASS_NAME_ , _BEHAVIOR_NAME_ , _PARAMETER_ ) ;
{ variation point = vpo_CLASS_NAME_ }
{ variation point = vpo_BEHAVIOR_NAME_ }
{ variation point = vpo_PARAMETER_ }
Return result;
```

_CLASS_NAME_ : 가변성 클래스 식별자
_BEHAVIOR_NAME_ : 가변성 클래스 내의 특정 오퍼레이션 식별자

그림 6 메타정보 기반의 가변부 선택 및 실행 코드

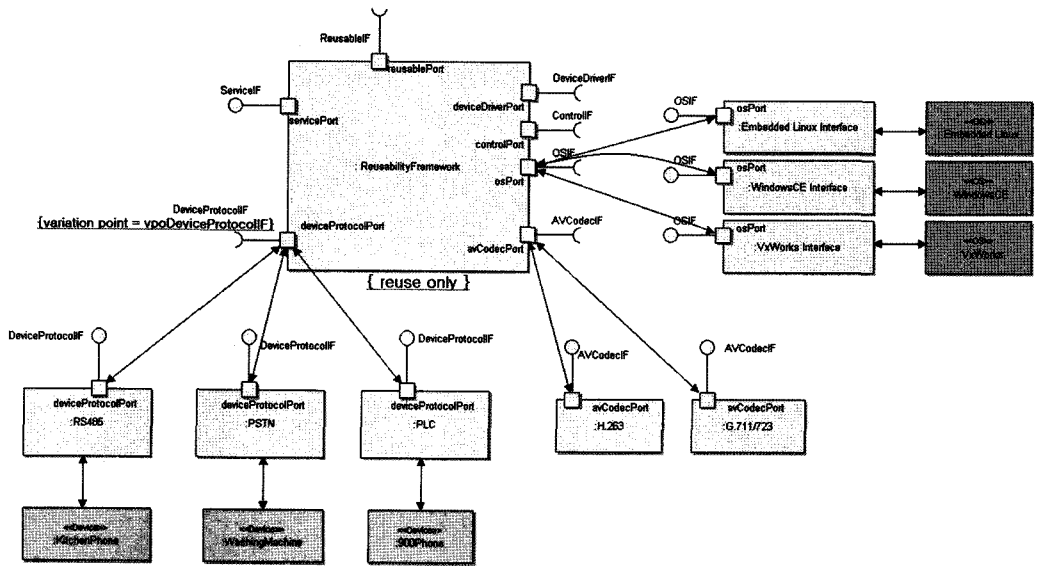

그림 7 가변성 어댑터에 의한 가변부 구조

경되어 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 8과 같이 도메인 요구사항(Domain A, B)에 따라 단일 어댑터('DeviceProtocolAdapter')을 통해 가변부 클래스를 선택하거나 다중의 가변부 클래스를 동시에 선택하여 서비스를 이용할 수 있다. 홈 네트워크 시스템의 경우 동시에 다양한 가전 디바이스에 신호를 전달해야 하는 경우에 이러한 가변적인 서비스 구조가 요구된다. 도메인 C와 같이 서로 다른 형태의 가변성 서비스를 요구하는 경우 다중의 가변성 어댑터를 동시에 이용할 수 있어야 한다. 홈 네트워크 시스템에서 다중의 가전 디바이스를 디바이스 프로토콜뿐만 아니라 가전 디바이스에 따른 멀티미디어 서비스를 제공하기 위해 AV 코덱을 가변적으로 접근하여 이용할 수 있다. 이러한 다중 어댑터를 통한 다중 기능의 제공이 기존 연구

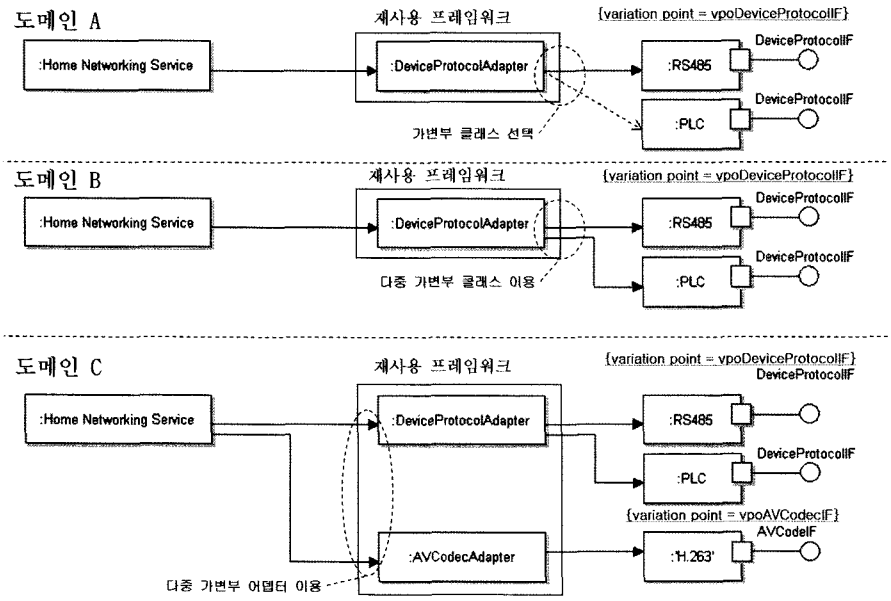

그림 8 도메인 요구사항에 따른 가변성 어댑터 구조

와 차별화된 특징이라고 할 수 있다.

• 동적 전개기

동적 전개기는 홈 네트워크 시스템 내부에서 가변성을 제공할 수 없는 경우 시스템 외부에서 요구하는 기능 클래스를 시스템 내부로 플러그하기 위한 도구이다. 플러그의 개념은 시스템 패키지 내에 요구하는 클래스를 포함하는 것이 아니라 시스템 운영 환경에 맞게 객체가 생성(Instantiate)되는 것을 의미한다.

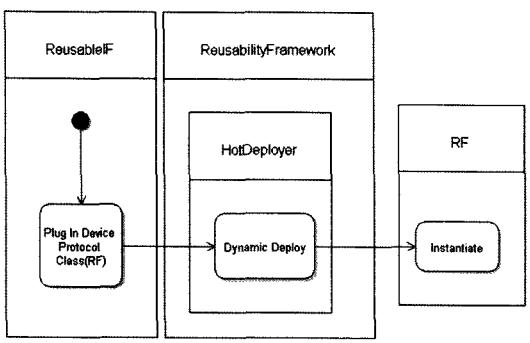

그림 9 동적 전개기에 의한 가변부 클래스 플러그 인

그림 9와 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.

• 자원 할당기

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

3.3 재사용 프레임워크에 의한 홈 네트워크 임베디드 시스템의 가변성 설계

본 논문에서 홈 네트워크 임베디드 시스템의 가변부를 변경하기 위한 기법은 재사용 프레임워크를 통해 가변부를 선택하는 기법과 임베디드 시스템 외부에서 요구하는 가변부를 플러그-인 하는 기법으로 구분하여 제안한다.

3.4 선택 기법

가변부 선택 기법은 홈 네트워크 임베디드 소프트웨어 내부에 존재하는 가변부 클래스 중에 하나의 클래스를 선택하여 가변부를 변경하는 기법이다.

그림 10은 홈 네트워크 임베디드 시스템에서 가변부 선택 메커니즘을 제공하기 위한 행위 흐름을 나타낸다. 재사용 인터페이스('ReusableIF')에서 디바이스 프로토콜을 설정하면 재사용 프레임워크는 디바이스 프로토콜에 대한 메타정보를 설정하고 설정된 디바이스 프로토콜을 초기화(Initialization)를 한다. 그림 10에서와 같이 동시에 다양한 디바이스 프로토콜을 설정하여 홈 네트워크 시스템이 동시에 다양한 디바이스 프로토콜을 사용할 수 있도록 제공할 수 있다.

가변부 선택 기법의 설정 및 서비스에 대한 상세한

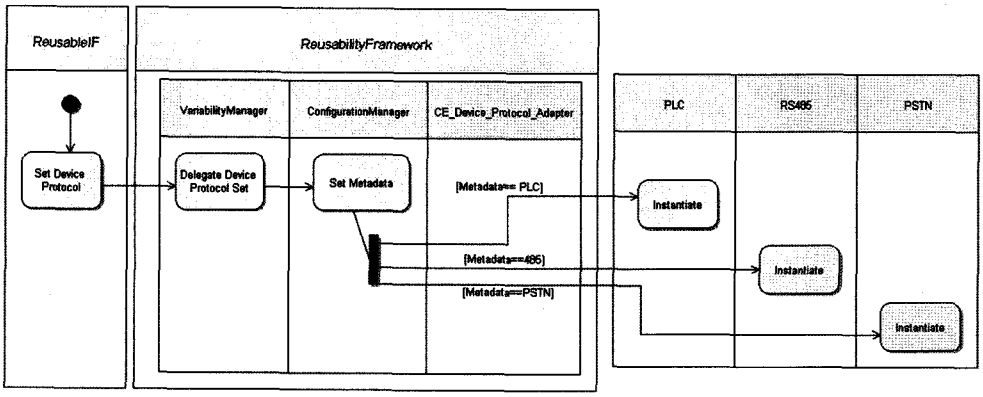

그림 10 디바이스 프로토콜 가변성에 대한 설정 행위 흐름

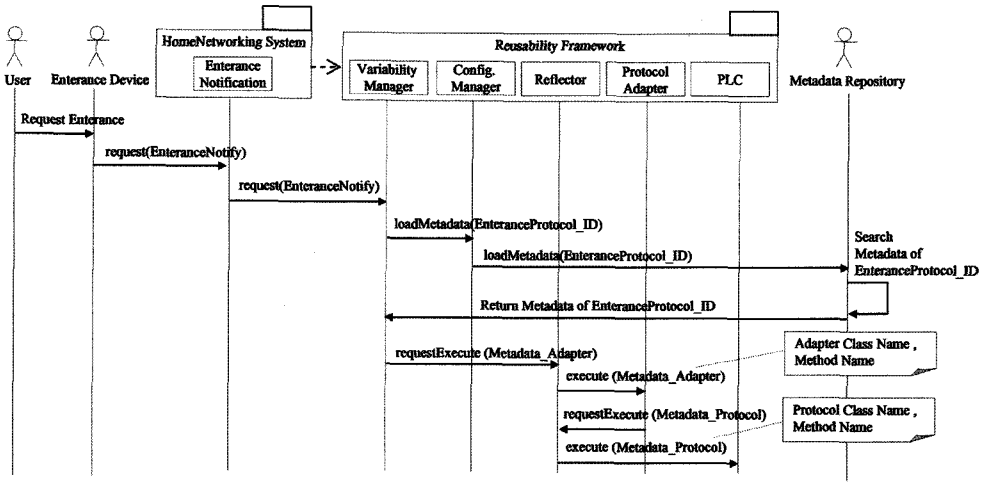

그림 11 가변부 선택 기법 순차도

흐름은 그림 11과 같으며 각각의 단계에 대해 정의한다.

(1) 가변부 메타정보 설정

그림 11과 같이 임베디드 시스템의 관리자는 가변부에 대한 메타정보를 설정한다. 관리자는 변경하려는 특정 가변부에 대한 시스템 명세를 통해 인터페이스 및 클래스들의 정보를 설정한다. 관리자는 XML 형태의 메타정보 저장소에 직접 접근하여 설정할 수 있으며 그림 5과 같은 메타정보를 설정된다.

(2) 서비스 요청

사용자는 임베디드 시스템의 서비스를 요청한다. 요청 서비스가 가변부를 호출하고 있으면 그림 11과 같이 재사용 프레임워크를 호출한다. 가변부를 사용하는 클래스는 가변부의 메타정보를 얻을 수 있는 가변부 식별자를 정의하여 호출한다(그림 6 참조).

(3) 재사용 프레임워크(가변성 관리기) 호출

가변부를 사용하는 영역은 가변부를 호출하기 위한 재사용 프레임워크의 가변성 관리기를 호출한다. 가변성 관리기는 가변부를 사용하는 클래스에서 전달받은 가변부 식별자를 기반으로 가변부의 메타정보를 호출하며 리플렉터를 통해 가변성 어댑터를 호출한다.

(4) 메타정보 호출

가변부 식별자를 통해 메타정보 저장소 내의 메타정보를 호출한다. 해당 메타정보는 가변부를 연결하는 가변성 어댑터, 실제적인 실행을 수행하는 클래스와 행위(오퍼레이션), 컨텍스트 정보를 포함하고 있다. 그림 5와 같이 가변부의 메타정보는 가변부의 식별자('_VARIABILITY_NAME_')를 기준으로 하나의 클래스 및 여러 클래스를 호출할 수 있다. 호출된 메타정보는 가변성 관리기에서 물리적 클래스를 실행할 수 있도록 전달된다.

(5) 가변성 어댑터 선택 및 실행

가변성 관리기는 호출된 메타정보를 기반으로 가변부를 연결할 수 있는 어댑터를 선택 및 실행한다. 어댑터는 가변부의 클래스에 대한 인터페이스를 통해 접근하므로 다른 클래스로 변경되더라도 영향을 받지 않는다.

그림 6과 같이 어댑터는 실행하려는 가변부 클래스의 메타정보를 호출한다.

(6) 가변 클래스 선택 및 실행

어댑터는 메타정보에 선택된 가변부 클래스의 행위(오퍼레이션)를 호출한다. 클래스 행위 실행 시 입력 데이타는 가변부 사용 클래스로부터 전달되며 출력 데이타는 재사용 프레임워크를 통해 가변부를 사용하는 클래스로 전달된다.

(7) 가변부 변경

가변부를 변경하고자 할 때 (1) 단계의 가변부 메타정보를 변경하여 (2)~(7) 단계를 실행한다. (1) 단계의 메타정보가 변경되더라도 가변부 사용 영역과 가변부는 전혀 변경이 없다. 단지 가변부에 대한 설정 메타정보만 변경된다.

본 논문에서 제안하는 재사용 프레임워크의 차별점은 단일 인터페이스에 대해 구현 클래스를 변경하는 제한적인 변경이 아니라, 다른 형태의 인터페이스로 변경하는 것이 가능하며 또한 단일에서 다중 인터페이스로 변경이 가능하다. 재사용 프레임워크는 메타정보에 따라 단일 가변 어댑터를 통해 가변성을 처리하거나 다중 어댑터를 호출하여 가변성을 처리할 수 있다. 이러한 기법을 통해 홈 네트워크 시스템의 요구사항인 하나의 이벤트로부터 서로 다른 디바이스를 동시에 제어할 수 메커니즘을 제공할 수 있다.

3.5 플러그 인 기법

가변부 플러그 인 기법은 임베디드 시스템 내부에서 가변 요구사항을 제공할 수 없을 경우에 임베디드 시스템 외부에서 가변 클래스를 시스템 내부로 플러그-인 하는 기법이다.

가변부 플러그 인 기법의 설정 및 서비스에 대한 상세한 흐름은 그림 12와 같다.

(1) 가변성 어댑터 및 클래스 플러그-인

그림 12와 같이 객체(Object) 타입을 입력 매개변수로 하는 동적 전개기 기능에 외부 가변성 어댑터 및 클래스를 입력한다. 재사용 프레임워크 내부에서 동적 전개기는 동적으로 플러그 인 된 가변성 어댑터 및 클래스를 활성화(Activation) 시킨다.

(2) 가변부 메타정보 설정

설정하는 메타정보는 가변부 선택 기법과 동일하며 플러그-인 경우, 변경하려는 가변부의 명세에 정의되어 있지 않으므로 새로운 클래스 메타정보를 설정한다. 설정되는 메타정보는 신규 어댑터 및 신규 클래스, 오퍼레이션의 식별자이다.

(3) 서비스 요청

사용자는 임베디드 시스템의 서비스를 요청한다. 요청 서비스가 가변부를 포함하고 있으면 그림 12와 같이 재사용 프레임워크를 호출한다. 임베디드 소프트웨어의 가변부 사용 클래스에 정의되는 가변부 식별자는 가변부가 임베디드 소프트웨어 내부 클래스(하드웨어 드라이버) 또는 외부 클래스 인지를 고려하지 않는다.

(4) 재사용 프레임워크(가변성 관리기) 호출

가변부를 사용하는 클래스는 가변부를 호출하기 위한 재사용 프레임워크의 가변성 관리기를 호출한다. 가변성 관리기는 가변부 식별자를 기반으로 가변부의 메타정보를 호출하며 리플렉터를 통해 가변성 어댑터를 호출한다. 이때 호출되는 어댑터는 재사용 프레임워크 내의 홈 네트워크 임베디드 시스템을 위한 가변성 어댑터이거나 외부 어댑터 일 수 있다.

(5) 메타정보 호출

가변부 식별자를 통해 메타정보 저장소 내의 메타정보를 호출한다. 가변부 식별자는 가변부가 변경되더라도 가변부 사용 클래스에서 사용되는 한 변경되지 않는다. 호출되는 메타정보는 가변부를 나타내는 어댑터, 클래스, 행위(오퍼레이션), 컨텍스트 정보와 같이 가변부 선택 기법과 동일하지만 임베디드 시스템의 외부 클래스에 대한 메타정보이다. 가변부 플러그 인 기법을 위한 가변부의 메타정보는 가변부의 식별자를 기준으로 단일 클래스 및 다중 클래스를 호출하며 임베디드 시스템 내

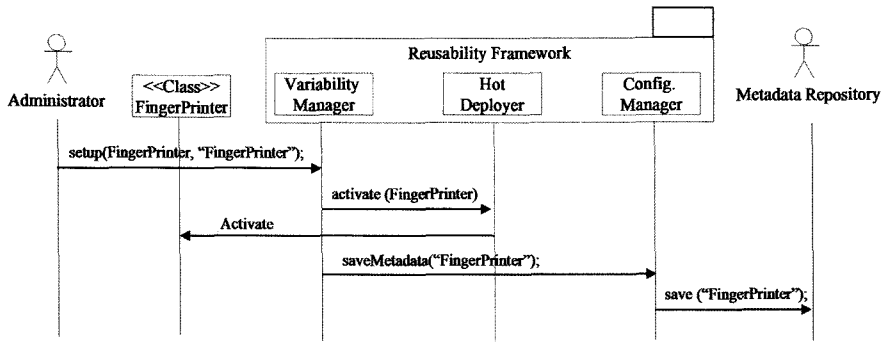

그림 12 플러그 인 기법 순차도

부와 외부 클래스를 조합하여 구성할 수 있다.

(6) 가변성 어댑터 선택 및 실행

가변성 관리기는 호출된 메타정보를 기반으로 가변부를 연결하는 어댑터를 선택 및 실행한다. 어댑터는 가변부의 클래스에 대한 인터페이스를 통해 접근하므로 다른 클래스로 변경 되더라도 가변부를 사용하는 다른 영역은 영향을 받지 않는다. 플러그 인 기법인 경우 메타정보가 외부 어댑터로 설정되어 있다면 전혀 다른 클래스 타입(인터페이스)으로 변경하는 경우이다.

(7) 가변 클래스 선택 및 실행

가변성 어댑터는 메타정보에 선택된 임베디드 시스템의 내부 및 외부 클래스의 행위를 호출한다. 다중 클래스 호출 시 임베디드 시스템 내부 인터페이스를 따르는 클래스를 호출하거나 외부 어댑터에 의해 외부 클래스를 호출할 수 있다. 입력 및 결과 데이타는 재사용 프레임워크를 통해 전달된다.

4. 실험 및 평가

본 논문에서 제시하는 홈 네트워크 임베디드 시스템의 재사용 프레임워크의 재사용성을 검증하기 위해 기존의 설계 사례와 비교하여 재사용성이 향상됨을 증명한다.

4.1 사례 연구

본 실험에서 제시하는 사례는 홈 네트워크 시스템에서 방문자가 현관에 방문하여 도어폰(Door Phone)을 눌렀을 때 방문 통보를 댁내의 다양한 가전 디바이스(월패드(Wall Pad), 주방폰, 셋탑, 핸드폰, 등)에 알리는 기능이다. 이 때 댁내의 다양한 가전 디바이스는 개발되는 도메인에 따라 다르게 적용될 수 있기 때문에 가변적인 처리가 요구된다. 따라서, 기존의 객체지향 기법으로 설계한 사례와 본 논문에서 제안하는 재사용 프레임워크를 사용하여 설계한 사례를 비교 검증한다.

그림 13은 홈 네트워크 솔루션의 출입 방문 시 'Visiting' 클래스와 가전 디바이스 제어 프로토콜인 'RS485' 클래스와 관계를 통해 주방폰에 방문자가 도착했음을 통보한다. 본 사례는 'Visiting' 클래스와 'RS485' 클래스가 직접적으로 결합되어 설계되므로 확장성이 크지 않다.

출입통보 설계 사례 2는 그림 14와 같이 'Visiting' 클래스와 가전 디바이스 제어 프로토콜인 'RS485' 클래스와 직접 연결되지 않으며 인터페이스를 통해 연결되도록 설계되었다. 프로토콜은 확장될 수 있더라도 다중의 프로토콜로는 확장이 불가능하다.

본 논문에서 제안하는 'Reusability Framework'을 이용한 설계 사례는 그림 15와 같으며 'Visiting' 클래스와

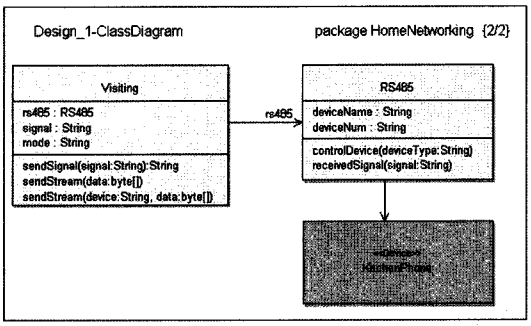

그림 13 출입통보 설계 사례 1

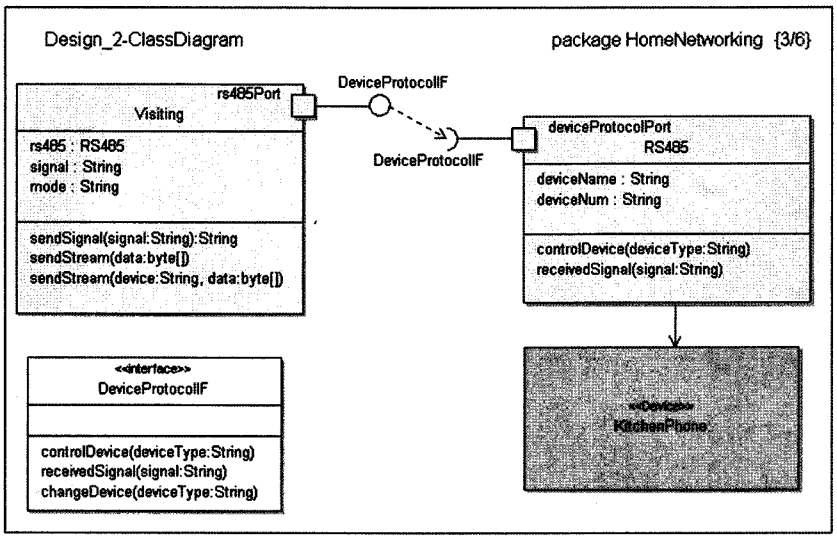

그림 14 출입통보 설계 사례 2

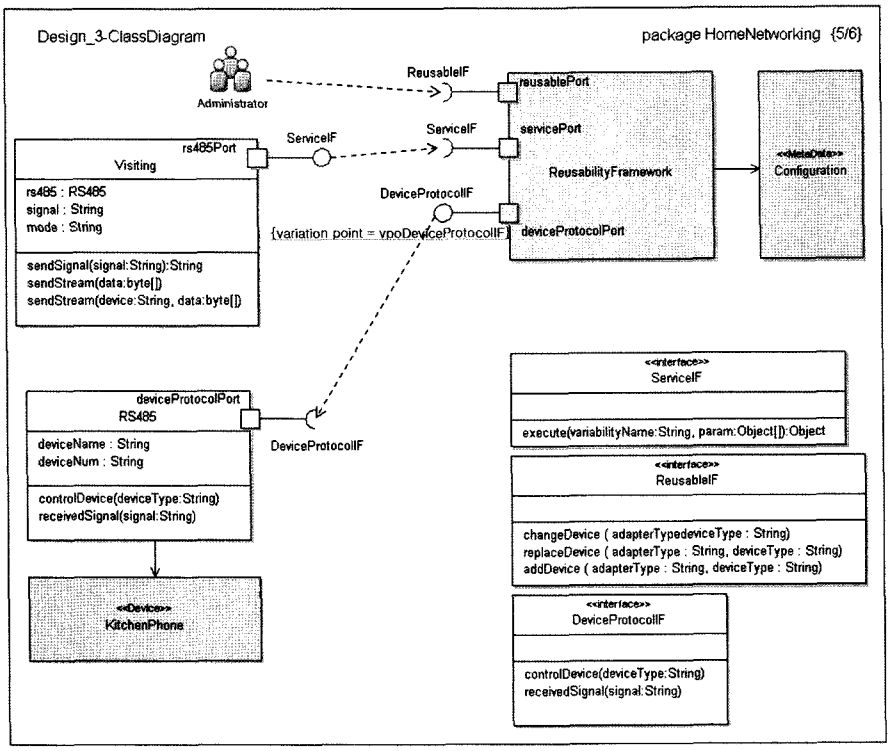

그림 15 출입통보 설계 사례 3 (Reusability Framework 이용)

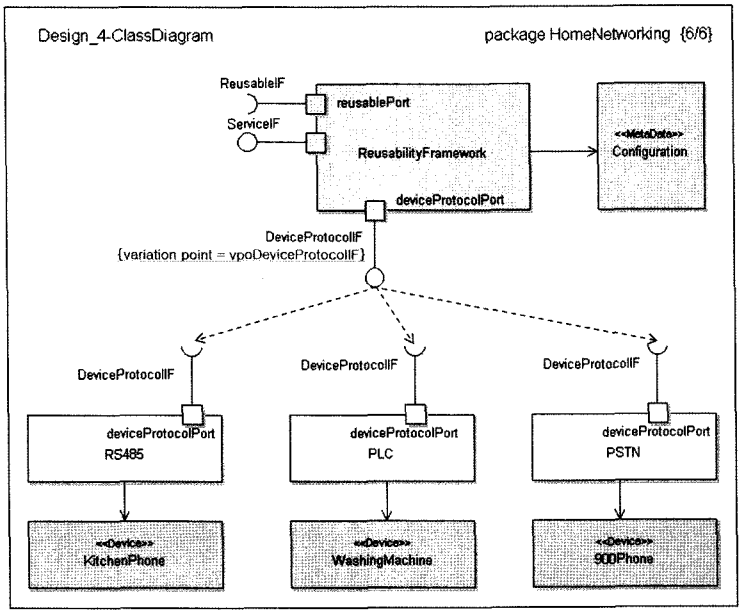

그림 16 다양한 디바이스 프로토콜 지원하기 위한 설계

가전 디바이스 제어 프로토콜인 'RS485' 클래스가 'Reusability Framework'를 통해 연결된다. 'Visiting' 클래스는 디바이스 제어 프로토콜 'RS485' 클래스와 직접 연결되지 않는다.

그림 16과 같이 'Reusability Framework'을 통해 다양한 디바이스 프로토콜을 지원할 수 있다. 'RS485',

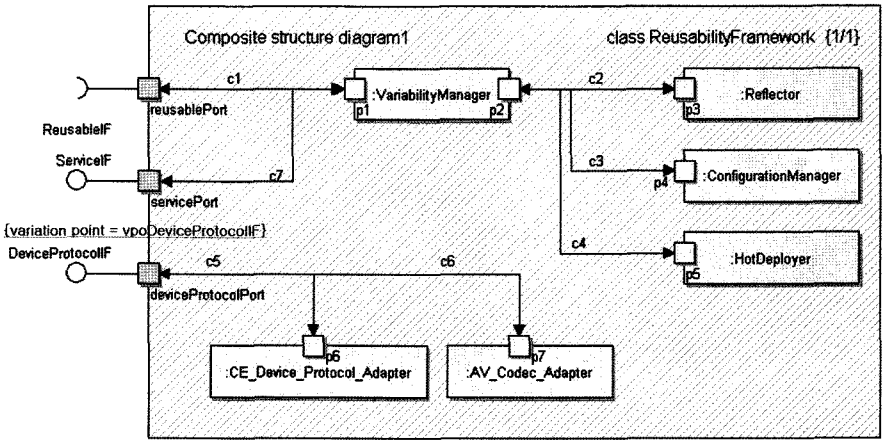

그림 17 Reusability Framework의 내부 구조(Composite Structure Diagram)

'PLC', 'PSTN' 프로토콜 클래스는 'DeviceProtocolIF' 인터페이스를 따르며, 'Reusability Framework'에서 다양한 프로토콜(PLC, PSTN, RF, 등)을 가변적으로 처리할 수 있다.

그림 17은 '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 어댑터이며, 'AV_Codec_Aapter' 클래스는 다양한 코덱을 제공하기 위한 어댑터 이다. 어댑터는 메타정보를 통해 단일 또는 다중으로 선택할 수 있다.

4.2 평가

본 평가에서는 Hironori Washizaki가 제시하는 평가 메트릭[24]을 기반으로 기존의 객체지향 설계 방식과 본 논문에서 제시하는 재사용 프레임워크를 이용한 설계 사례를 평가한다. 재사용 평가 메트릭은 그림 18과 같이

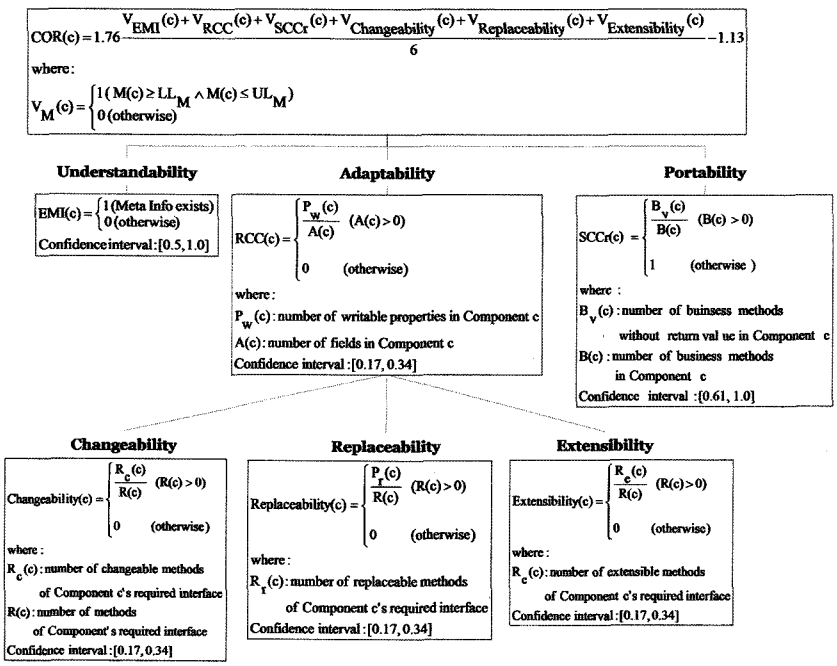

그림 18 재사용 평가 메트릭

Factor	Measured Value		
	Case 1	Case 2	Case 3
Meta Info	No	No	Exists
Number of Field	5	5	5
Number of Writable Properties	3	3	3
Number of Business Method	5	5	5
Number of Business Method without return value	4	4	4
Number of Method of Required Interface	0	3	3
Number of Changeable Method of Required Interface	0	1	1
Number of Replaceable Method of Required Interface	0	0	1
Number of Extensible Method of Required Interface	0	0	1

Metric	Measured Value			V_{Metric}	Measured Value		
	Case 1	Case 2	Case 3		Case 1	Case 2	Case 3
EMI	0	0	1	V_{EMI}	0	0	1
RCC	3/5=0.6	3/5=0.6	3/5=0.6	V_{RCC}	0	0	0
SCC_r	4/5=0.8	4/5=0.8	4/5=0.8	V_{SCCr}	1	1	1
Changeability	0/0=0	1/3=0.3	1/3=0.3	$V_{changeability}$	0	1	1
Replaceability	0/0=0	0/3=0	1/3=0.3	$V_{replaceability}$	0	0	1
Extensibility	0/0=0	0/3=0	1/3=0.3	$V_{Extensibility}$	0	0	1

Case 1 $$COR = 1.76 \times \frac{0+0+1+0+0+0}{6} - 1.13 = -0.84$$

Case 2 $$COR = 1.76 \times \frac{0+0+1+1+0+0}{6} - 1.13 = -0.54$$

Case 3 $$COR = 1.76 \times \frac{1+0+1+1+1+1}{6} - 1.13 = 0.34$$

그림 19 재사용 측정값

이해성(Understandability), 적응성(Adaptability), 변경성(Changeability), 교체성(Replaceability), 확장성(Extensibility), 이식성(Portability)을 평가한다.

각 메트릭에 대한 정의는 다음과 같다.

정의 1. 이해성(EMI: Existence of Meta-Information): EMI(c)는 한 컴포넌트 내에 메타 정보의 존재 유무를 의미한다.

정의 2. 컴포넌트 적응성(RCC: Rate of Component Adaptability): RCC(c)는 한 컴포넌트 내의 속성들 가운데 수정 가능한 속성들의 비율을 의미한다.

정의 3. 컴포넌트 이식성(SCCr: Rate of Component Portability): SCCr(c)는 한 컴포넌트 내의 비즈니스 메소드들 가운데 리턴 값이 없는 비즈니스 메소드들 수의 비율을 의미한다.

정의 4. 컴포넌트 변경성(Changeability): Changeability(c)는 한 컴포넌트 내의 요구 인터페이스들의 메소드들 가운데 변경 가능한 메소드들의 비율을 의미한다.

정의 5. 컴포넌트 교체성(Replaceability): Replaceability(c)는 한 컴포넌트 내의 요구 인터페이스의 메소드들 가운데 교체 가능한 메소드들의 비율을 의미한다.

정의 6. 확장성(Extensibility): Extensibility(c)는 컴포넌트의 요구 인터페이스의 메소드들 가운데 확장 가능한 메소드들의 비율을 의미한다.

재사용 메트릭을 이용하여 측정한 사례 연구(Case 1, Case 2, Case 3)의 측정 결과는 그림 19와 같다.

기존 설계 방식인 설계 사례 1과 설계 사례 2와 다르게 'Reusability Framework'을 이용한 설계 사례는 COR이 0 보다 크므로 재사용성이 있다고 할 수 있다[24]. 구조적 측면에서는 'Reusability Framework'이 추가되기 때문에 복잡성이 높아질 수 있으나 설계 사례 1과 설계 사례 2에 비해 변경성, 교체성, 확장성이 좋아지므로 재사용성이 향상된다고 할 수 있다.

5. 결 론

홈 네트워크 시스템은 다양한 도메인의 요구사항에 따라 다양한 디바이스를 지원할 수 있어야 한다. 본 논문에서는 홈 네트워크 시스템의 이러한 가변적인 처리를 지원하기 위해 홈 네트워크 임베디드 소프트웨어 내에 재사용 프레임워크를 제안한다. 홈 네트워크 시스템의 디바이스 변경에 따라 제어, 디바이스 프로토콜, 디바이스 드라이버, 운영체제, AV 코덱 등이 변경되어야 하며, 재사용 프레임워크에서는 이러한 가변적인 부분을 인터페이스나 동적으로 처리될 수 있는 메커니즘을 제공한다. 홈 네트워크 시스템의 가변적인 처리 기법은 홈 네트워크 시스템 내의 여러 기능 중에 하나를 선택하는 선택 기법과 홈 네트워크 시스템 외부에서 새로운 기능을 제공하는 플러그 인 기법이 있다. 두 기법은 메타정보를 통해 설정할 수 있으며 개발 시점뿐만 아니라 운영 시점에도 다양한 요구사항을 지원할 수 있다. 본 논문에서 제안하는 홈 네트워크 임베디드 시스템 재사용 기법에 대한 사례연구를 통해 다양한 홈 네트워크 도메인에 재사용될 수 있음을 증명하였으며, 향후 연구에서는 다양한 임베디드 시스템의 재사용성을 향상시킬 수 있는 공통의 재사용 프레임워크 기법을 연구할 것이다.

참 고 문 헌

[1] Selic B., Gullekson G., and Ward P.T., Real-Time Object-Oriented Modeling, John Wiley&Sons, 1999.
[2] Comma H., "A Software Design Method for Real-Time Systems," Communications of ACM, Vol.27, No.7, pp. 938-949, Sept. 1984.
[3] David E. S., An Embedded Software Primer, Addison Wesely, 1999.
[4] Raj K., Embedded Systems: Architecture, Programming and Design, McGraw Hill, 2004.
[5] Ready J. and Howard D., "Structuring Real-Time

Application Software Part1," VMEbus Systems, pp. 33-45, April, 1991.
[6] Axel J., Modeling Embedded System and SOCs, Mogan Kaufmann, 2004.
[7] Jose C., "Next-Generation Object-Oriented Software Analysis and Design Methodology," at URL: http://www.hpl.hp.com/fusion/ma_961007.html, 1996.
[8] Szyperski C., Component Software: Beyond Object-Oriented Programming, Addison-Wesley, 2002.
[9] Kang K., "Issues in Component-Based Software Engineering," International Workshop on Component-Based Software Engineering 1999.
[10] Hopkins J., "Component Primer," Communication of the ACM Vol.43, No.10, October 2000.
[11] Short K., Component Based Development and Object Modeling, Sterling Software, Technical Handbook Version 1.0, February 1997.
[12] Coplien J., Hoffman D., and Weiss D., "Commonality and Variability in Software Engineering," IEEE Software, pp. 37-45, November 1998.
[13] Weiss D. M., "Commonality Analysis: A Systematic Process for Defining Families," Second International Workshop on Development and Evolution of Software Architectures for Product Families, February 1998.
[14] Kang, K. C., Cohen, S. G., Novak, W. E. and Peterson, A. S., "Feature-oriented Domain Analysis(FODA) Feasibility Study," Technical Report CMU/SEI-90-TR-21, Software Engineering Institute (SEI), November 1990.
[15] Gamma E., et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.
[16] Pree, W., Design Patterns for Object-Oriented Software Development. Addison-Wesley, 1995.
[17] I. Sora, P. Verbaeten, and Y. Berbers, "Using Component Composition for Self-Customizable Systems," Workshop on Component-Based Software Engineering, ECBS 2002, April 8-11, 2002, Lund, Sweden.
[18] R. P. e Silva, et al., "Component Interface Pattern," Procs. Pattern Languages of Program, 1999.
[19] 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), November 2000.
[20] Bachmann F., Bass Len, http://www.sei.cmu.edu/plp/variability.pdf, "Managing Variability in Software Architecture," Software Engineering Institute (SEI), 2001.
[21] Diana L. and Gomaa H., "Modeling variability in software product lines with the variation point model," Science of Computer Programming 53, 2004.
[22] Java Developer Network, http://java.sun.com.
[23] Microsft Developer Network, http://msdn2.microsoft.com.
[24] 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.

김 철 진

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

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. 관심분야는 CBSE, Embedded Software, Service-Oriented Computing, SOA