임베디드 소프트웨어 개발을 위한 제품계열 중심의 개발프로세스 모델 설계

(A Design of Development Process Model of Product Lines for Developing Embedded Software)

홍 기 삼 † 윤 희 병 ††
(Kisam Hong) (Heebyung Yoon)

요 약 최근 임베디드 소프트웨어가 다양한 분야에서의 개발이 두드러지면서 그 요구사항들도 다양해지고 있다. 그 중 이슈화되고 있는 것 중 하나가 하드웨어와의 종속성을 반영한 체계적인 개발방법을 제시하는 것이다. 기존의 개발방법들은 하드웨어와의 밀접한 관계, 여러 유사 도메인에 대한 고수준의 재사용성 요구 등 임베디드 소프트웨어가 갖고 있는 특성들을 효과적으로 반영하지 못하고 있다. 따라서 본 논문에서는 제품계열적 접근방법을 통해 효율적인 임베디드 소프트웨어 개발방법을 제시하기 위한 개발프로세스 모델 설계방법을 제안한다. 개발프로세스 모델의 설계 중점은 먼저 효율적 요구사항 도출기법으로 디렉토리 방식의 도메인 스코핑 방식과 IDEF0 기반 비즈니스 모델을 제안한다. 다음으로 서비스 구조 기반의 컴포넌트 도출방식과 하드웨어 종속성을 고려한 아키텍처 설계 방식을 제안한다. 마지막으로 제안한 개발프로세스 설계모델이 어떻게 임베디드 소프트웨어 개발에 적용되는지를 보이기 위해 다중센서데이터 융합시스템에 적용한 결과를 설계과정마다 제시한다.

키워드 : 임베디드 소프트웨어, 제품계열, 핵심자산, 비즈니스 모델링, 아키텍처 최적화

Abstract Recently, the requirements of the embedded software are getting diverse as the diversity of embedded software application fields increases. The systematic development methods are issued to deal with the dependency between hardware and software. However, the existing development methods have not considered the software's close connection to hardware and the high-level reusability for common requirements of several similar domains. In this paper, we propose a design method of development process model of product lines to support an efficient development method for embedded software. For this, we firstly suggest a domain scoping method and an IDEF0(Integration DEFinition)-based business model for extracting the efficient requirements. Next, we present a component deriving method based on the service architecture and an architecture design method after considering the hardware dependency. And we explain the artifacts of MSDFS(Multi Sensor Data Fusion System) at each design step in order to show how the proposed model can be applied to the embedded software development.

Key words : Embedded Software, Product Line, Core Assets, Business Modeling, Architecture Optimization

1. 서 론

임베디드 소프트웨어는 전기가전, 국방 무기시스템, 의료정보시스템 등에 내장될 수 있도록 저렴한 가격, 소형화, 저전력 소비, 고신뢰성, 소프트웨어 기능 및 성능의 최적화, 하드웨어에 대한 효율적 자원관리 등 다양하고 복잡한 특징을 가지고 있다. 이러한 임베디드 소프트웨어는 1970년대부터 활발히 개발되어 오면서 단순하고 독립적인 소형시스템에 장착되어 운영되었던 과거와는 달리 최근에는 초정밀의 동작제어와 함께 복잡하고 다양한 플랫폼 환경을 요구하거나 다수의 소프트웨어를 통합하고 제어하는 시스템 종속적인 요구사항들이 많이 발생하고 있다.

일반적으로 임베디드 소프트웨어는 시스템의 하드웨어를 제어하는 하드웨어에 종속적인 부분(HP: Hardware

† 학생회원 : 국방대학교 전산정보학과
iamhongsam@hanmail.net
†† 종신회원 : 국방대학교 전산정보학과 교수
hbyoon37@hanmail.net
논문접수 : 2006년 3월 30일
심사완료 : 2006년 9월 25일

dependent Part), 운영체제에 종속적인 애플리케이션 부분(OP) 그리고 임베디드 시스템의 요구사항 자체를 구현한 애플리케이션에 종속적인 부분(AP)으로 분류된다. 이러한 임베디드 소프트웨어를 효율적으로 개발하기 위해서 최근에 이슈가 되고 있는 부분이 시스템에 종속적인 문제, 즉 하드웨어와의 종속성에 따른 설계문제를 해결할 수 있는 체계적인 개발방법을 제시하는 것이다.

하지만 기존의 임베디드 소프트웨어 개발방법들은 하드웨어와 밀접한 관계, 여러 유사 도메인에 대한 고수준의 재사용성 요구 등 임베디드 소프트웨어가 갖고 있는 여러 중요한 특징들을 제대로 반영하지 못함으로서 유동적이지 못하고 시스템 중복성을 유발하는 등 여러 비효율적인 결과들을 초래하였다.

이러한 비효율적인 요소들을 제거하기 위해 다양한 개발방법론들이 연구되었는데 이러한 방법들에는 제품계열적 개발방법론[1-3], 플랫폼 기반 개발방법론[4,5], 모델-주도 접근방법론[6] 등이 대표적이라 할 수 있다. 이 중에서 수요자의 기호 충족과 개발비용 절감을 위한 목적으로 도메인 수요를 목표로 하여 공통의 특성을 가진 제품에 대해 제품계열을 형성하고 제품계열의 가변성을 고려하여 설계한 다음 설계된 핵심자산을 활용하여 최종 제품에 대한 버전을 관리하는 제품계열적 개발방법론이 최근 임베디드 소프트웨어 개발을 위한 개발 프레임워크로서 대두되고 있다.

따라서 본 논문에서는 효율적인 임베디드 소프트웨어 개발을 위해 제품계열 중심의 핵심자산 개발프로세스 모델에 대한 효율적인 설계방법을 제안한다. 이를 위해 먼저 효율적 요구사항 도출기법으로 디렉토리 방식의 도메인 스코핑 방식과 IDEF0 기반 비즈니스 모델 설계 방식을 제안하며, 시스템 최적화 설계를 위해서는 서비스 구조 기반의 컴포넌트 도출방식과 하드웨어 종속성을 고려한 아키텍처 설계 방식을 제안한다. 그리고 제안한 개발프로세스 설계모델이 어떻게 임베디드 소프트웨어 개발에 적용되는지를 보이기 위해 다중센서데이타 융합시스템에 적용한 결과를 설계과정마다 제시한다.

2. 관련연구

2.1 제품계열개발방법론

소프트웨어 생산기술로 최근 각광받고 있는 제품계열 공학은 미국과 유럽에서 많이 연구되고 있다. 제품계열 공학은 크게 두 단계로 나눠 볼 수 있다. 즉 첫 번째 단계는 제품계열 내의 핵심자산(core asset)을 구축하는 단계이고, 두 번째 단계는 핵심자산으로부터 개별제품을 생산하는 단계이다. 기존의 컴포넌트 기반 개발방법과 다른 점은 재사용 자산을 도메인 분석과 제품 패밀리 분석을 통해 공통의 아키텍처로 구축한다는 것과 제품 생산과정에서 핵심자산의 공통성과 가변성을 이용하여 제품을 생산한다는 것이다[7].

제품계열 개발방법론에서 설계중점은 첫째 임베디드 S/W의 특징을 고려한다는 것이다. 즉 요구사항 도출시 임베디드 시스템 내에 일어나는 내부흐름인 입력, 출력, 제어 등을 중심으로 요구사항을 도출하고, 하드웨어 종속성도 고려하여 설계하며, 기능 수행에 최적화된 설계가 이루어져야 하는 것이다. 둘째로 도메인 모델(Feature 모델)에서 아키텍처 모델로의 전환에 대해 추적성 보장문제가 해결되어야 한다는 것이다. 즉 도메인 영역에서 도출된 Feature들을 체계적으로 아키텍처 모델에 반영하여 설계가 이루어져야 하며, 설계 도메인의 다변화로 인한 스코핑 단계에 체계적인 기법을 도입해야 한다.

2.2 다중센서데이타 융합시스템(MSDFS)

다중센서데이타 융합시스템(MSDFS: Multi Sensor Data Fusion System)은 여러 센서로부터 획득된 이질의 데이타를 정규화된 포맷으로 융합하여 단일 센서에서의 획득오차를 최소한으로 줄이고 표적의 정확한 식별과 판단을 지원하는 기반 데이타를 제공하도록 제작된 시스템이다[8]. 이러한 다중센서데이타 융합시스템은 여러 유사한 데이타를 처리하는 부분들로 구성되어 고수준의 재사용성(reusability)이 요구된다.

다중센서로부터 데이타를 융합하는 시스템은 많은 분야에서 활용되고 있는 핵심 소프트웨어이다. 즉 국방 분야에서는 육군의 방공자동화체계, 공군의 자동화방공체계, 해군의 협동교전능력체계의 핵심 소프트웨어이며, 민간 분야에서는 민간항공 관제시스템에 활용되고 있다. 따라서 MSDFS는 항적처리 시스템분야에서 제품계열화가 가능하고 또한 데이타 융합기능을 수행하는 시스템 분야에서도 제품계열화가 가능하다.

3. 핵심자산 개발프로세스 모델 설계

3.1 제안 개발프로세스 모델 개념

본 논문에서 제안한 핵심자산 개발프로세스 모델은 요구사항 도출기법 설계와 임베디드 시스템 최적화 설계에 초점을 둔다. 요구사항 도출은 디렉토리 방식의 도메인 스코프 방법을 제안하여 도메인 분석 영역을 효율적으로 정의하고 IDEF0모델 기반의 비즈니스 모델 설계를 통해 시스템 흐름에 종속적인 요구사항을 도출한다. 임베디드 시스템 최적화는 서비스 구조를 통한 컴포넌트 도출방법을 통해 도메인 영역분석에 식별된 Feature들을 아키텍처 모델로 반영할 수 있는 가이드라인을 제시하고, 하드웨어와 소프트웨어 종속성을 고려한 아키텍처 설계를 통해 시스템 최적화 방안을 제안한다. 본 논문에서 제안한 핵심자산 개발프로세서 모델 설계 과정이 그림 1에 나타나 있다.

그림 1에서 개발프로세스는 도메인 분석과 설계단계에 스코프 테이블, 그래프 산출물에 디렉토리 방식을 적용하고, 비즈니스 모델에는 IDEF0 기반의 모델기법을 적용하였으며, Feature 모델과 객체분석 모델에 서비스 구조를 적용하여 컴포넌트를 도출하였다. 또한 종속성 모델과 아키텍처 모델에는 하드웨어 종속성을 고려한 아키텍처 모델기법을 정의하여 임베디드 특성을 고려하였다.

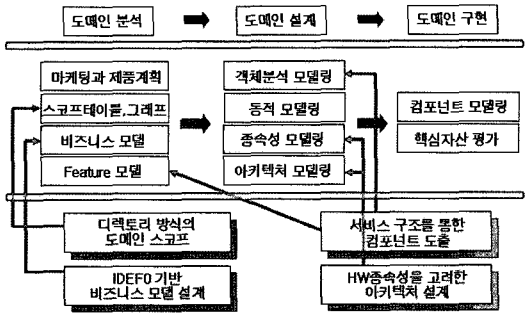

그림 1 핵심자산 개발프로세스 모델 설계과정

3.2 효율적 요구사항 도출

3.2.1 디렉토리 방식의 도메인 스코핑

최근 시스템 간의 기술적 벽이 허물어져 서로간의 컨버전스 경향이 두드러짐에 따라 영역정의 활동이 모호해지고 통합단계에서 요구사항이나 시스템적 변화가 발생할 경우 소프트웨어뿐만 아니라 시스템 전체를 변경해야 하는 치명적 문제가 발생하고 있다. 따라서 임베디드 소프트웨어 설계에서는 도메인 분석단계에서 도메인 영역을 정의하는 스코프가 요구사항 도출의 핵심요소로 대두되고 있다. 이에 본 논문에서는 도메인 정의단계의 체계적 기법으로 디렉토리 방식의 도메인 스코핑 방법을 제안한다.

디렉토리는 유사속성을 가진 객체, 요소들의 풀(pool)로서 미리 정의해 둔 공간을 말한다. 따라서 개발대상 체계를 분리원칙에 의해 세분화하고 검증하여 객체화한 후 이를 미리 정의된 디렉토리 풀 안으로 분류하여 도메인을 정의하며 최종적으로 시스템 특성에 따라 선정된 디렉토리명을 변경함으로써 스코프를 정의한다. 이러한 도메인 스코핑 과정이 그림 2에 나타나 있다.

제안된 스코핑 과정은 후보 스코프 도출, 후보 스코프 검증, 스코프 디렉토리 선정의 세 단계로 구분된다. 먼저 후보 스코프 도출은 임베디드 시스템 속성인 성능, 확장성, 트랜잭션, 재사용성, 개발의 직관성 등을 분리원칙으로 선정하여 개략적 시스템을 분리하는 단계로서 개발자 및 사용자 간의 의사소통을 통해 파레토법칙(Pareto's law) 안에서 이루어진다[9]. 즉 개발자와 사용자 간에 5개의 분리원칙에 의해 그 특성을 대표할 수

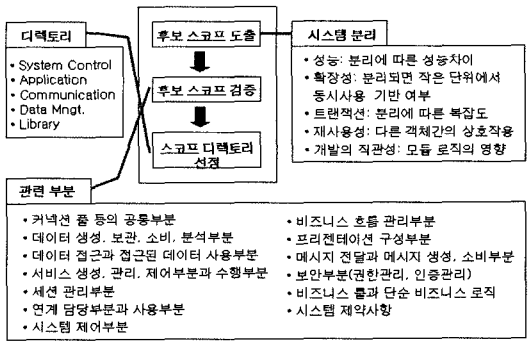

그림 2 디렉토리 방식의 도메인 스코핑 과정

있는 20%의 디렉토리를 선정한다는 것이다.

후보 스코프 검증단계에서는 모듈분리 기준을 검증기준으로 적용하여 후보 스코프에 대한 검증 및 스코프가 시스템에 어떤 부분과 관련되어 있는지 지정한다. 스코프 디렉토리 선정단계에서는 검증단계의 관련부분과 디렉토리명 간의 상관관계를 표 1과 같이 정의하고 관련 부분을 기준으로 디렉토리 풀에 후보 스코프를 분류한다.

표 1 디렉토리명과 관련 부분과의 상관관계

디렉토리	관련 부분
System Control	비즈니스 흐름관리, 시스템 제어부분
Application	서비스 생성/관리/제어 부분과 수행부분, 프리젠테이션 구성 부분, 비즈니스 규칙과 로직 부분
Communication	커넥션 풀 등의 공통부분, 세션관리 부분, 연계담당 부분, 메시지 전달/생성/소비 부분
Data Management	데이타 접근과 사용부분, 데이타 생성/보관/소비/분석 부분
Library	보안과 인증부분, 시스템 제약사항

예 1) 디렉토리 방식의 도메인 스코핑 과정을 MSDFS에 적용하여 스코프 그래프를 도시한다.

MSDFS에 대한 스코프 그래프를 도시하기 위해서는 먼저 MSDFS 센서와 내부 기능들을 Inside와 Outside로 구분하고 트랜잭션 처리, 확장성, 재사용성 측면을 고려해 스코프명을 도출한다. 그런 다음 표 1의 상관관계를 이용하여 도출된 스코프명 각각에 대해 디렉토리를 식별하며, 마지막으로 식별된 디렉토리를 구성원(후보 스코프)의 특성을 고려해 재명명하여 작성한다. 이렇게 하여 작성된 스코프명을 Application, System Control, Data Management, Communication, Library 등의 디렉토리별 계층 순으로 재구성하고 그런 다음 이를 도표화시켜 그림 3과 같이 도시한다.

3.2.2 IDEF0 기반 비즈니스 모델 설계

비즈니스 모델은 개발대상 시스템에 대한 비즈니스

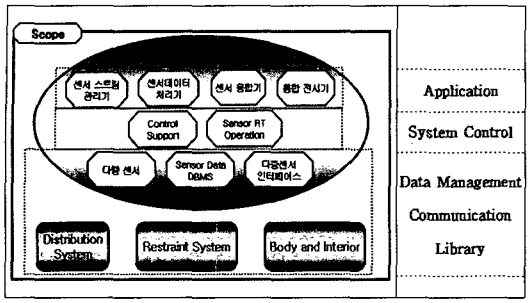

그림 3 MSDFS의 스코프 그래프

흐름을 구체적으로 파악하기 위하여 비즈니스 조직과 프로세스 흐름을 정의하고 비즈니스 객체와 이들 간의 연관관계를 정의한다. 이러한 비즈니스 모델은 UML과 IDEF0 모델을 통해 정의될 수 있다. IDEF(ICAM Definition)는 1980년대 초 미 공군에 의해 시스템을 기능, 정보, 동적관점에서 모델링하기 위한 방법으로 개발되어 임베디드 소프트웨어의 특성을 표현하는 설계분야에 널리 사용되고 있다. IDEF는 최상위 레벨을 나타내는 A0에서 A1, A2와 같은 세부 프로세스로 나뉘는 분할과정을 거쳐 아키텍트가 원하는 수준까지의 액티비티를 나타내고 그에 따른 관계, 기능 및 제약사항을 표현한다. 표현은 다르지만 UML의 시퀀스, 상태 다이어그램 등도 처리와 흐름 그리고 예외라는 다이어그램이 갖는 정보는 유사할 수 있고, 고수준의 프로세스를 나타냄으로써 IDEF보다 더 많은 정보를 내재할 수 있다. 하지만 임베디드 소프트웨어와 같은 내부처리와 흐름이 중요시되는 소프트웨어의 경우 IDEF가 보다 널리 사용된다. 국방무기체계 개발을 위해 2005년 제정된 국방 CBD 개발방법론이 비즈니스 모델링 과정의 흐름도(액티비티 다이어그램)와 개념도(E-R 다이어그램)를 IDEF0로 표현하여 모델링하는 것을 선호하는 것이 일 예라 할 것이다.

따라서 요구사항 분석을 위한 효율적인 비즈니스 모델링 과정은 그림 4와 같이 기존의 4단계를 줄여 3단계로 조정할 수 있다. 즉 대상 체계의 조직흐름을 파악하여 그 결과를 입력값으로 IDEF0 다이어그램을 작성하고, 최종적으로 Feature 모델과 결정트리를 작성한다.

비즈니스 모델 기반의 요구사항 분석과정을 상세히

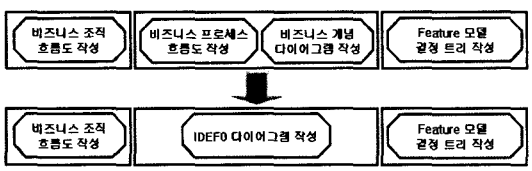

그림 4 비즈니스 모델링 과정

살펴보면 먼저 도메인 정의를 통해 도출된 스코프를 중심으로 요구사항을 도출하고 도출과정의 입력물로서 상위요구사항정의서와 협동다이어그램을 작성한다. 그런 다음 도출된 요구사항을 통해 IDEF0 비즈니스 모델과 요구사항명세서를 작성하게 된다. 최종적으로 도메인 분석단계의 Feature 모델과 Feature 결정트리를 작성한다.

제안한 요구사항 분석과정에서 IDEF0 비즈니스 모델과 요구사항명세서 간에는 반복적인 상호보완 관계가 계속적으로 이루어져야 한다. 즉 IDEF0 모델의 구성요소인 Input, Output, Control, Mechanism, Note는 비즈니스 단위인 액티비티가 수행하는 기능들에 대한 로직 흐름의 주체이기 때문에 요구사항명세서에 반영되어야 한다. 이를 위해 IDEF0 모델의 구성요소를 요구사항명세서의 관련 요구사항에 반영하고 요구사항 설명의 주어와 목적어로 적용한다.

예 2) IDEF0 비즈니스 모델 기반의 요구사항 분석과정을 MSDFS에 적용하여 Feature 모델을 작성한다.

Feature 모델은 CA(Capability), OE(Operation Environment), DT(Domain Technology), IT(Implementation Technique)의 계층구조로 구분하여 도출된 Feature 간 관계를 고려하여 그림 5와 같이 작성한다. 참고로 그림에 나와 있는 '데이타 융합'은 식별된 하나의 객체를 나타낸다.

3.3 시스템 최적화 설계

3.3.1 서비스 구조를 통한 컴포넌트 도출

서비스라는 개념은 최근 SOA(Service Oriented Architecture)를 통해 많은 연구 및 활용이 이루어지고 있다[11,12]. SOA에서 정의하는 서비스는 독립적인 비즈니스 기능을 구현한 소프트웨어 컴포넌트로서 메시지를 이용해 공개된 인터페이스를 통해 접근하는 네트워크 기능을 가진 소프트웨어 개체를 말한다.

본 논문에서는 이러한 서비스 개념의 독립적인 비즈니스 개념을 도입하고 서비스를 구조화하여 인스턴스화함으로써 객체 레벨의 최적화 방안을 제안한다.

제안에 앞서 서비스 구조를 Feature 모델에 적용하기 위해 SOA에서 사용되는 전형적인 서비스 구조(Service Fasade, Process Service, Business Entity Layer, Data Representation Layer, Data Layer)를 PLE(Product-Line Engineering)의 Feature 모델구조(CA, OE, DT, IT)와 유사한 매핑관계를 갖도록 그림 6과 같이 변경하였다. 그 결과 서비스 구조가 Service Fasade, Business Entity Layer, Process Layer, Data Layer로 재설정되었다. 단 이러한 재설정은 서비스 구조를 Feature 모델에 적용하기 위해 용어측면에서 변화를 준 것이고 구조적 측면에서는 기존의 서비스 구조와 동일한다.

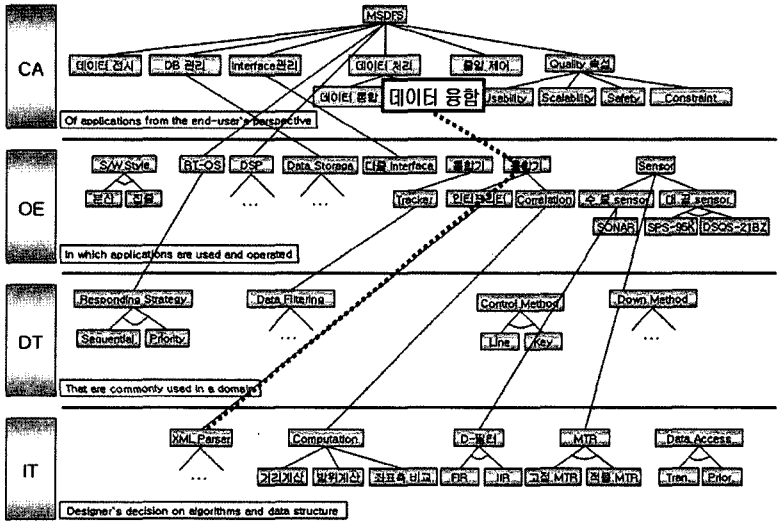

그림 5 MSDFS의 Feature 결정트리

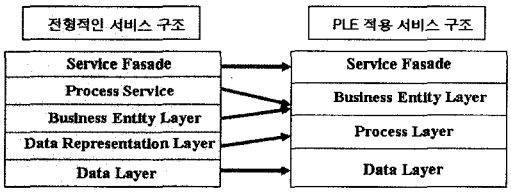

그림 6 PLE 적용 서비스 구조

PLE를 적용한 서비스 구조에서 Service Fasade는 서비스 개념에서 제어흐름과 공개 인터페이스 역할을 수행하며, Business Entity Layer는 요구사항 도출과정에서 식별된 비즈니스를 반영한다. 그리고 Process Layer와 Data Layer는 비즈니스 수행을 위한 처리기 및 처리된 데이타를 저장하기 위한 기능이다.

서비스 구조는 비즈니스 모델의 비즈니스 단위와 Feature 결정트리에서 관련 Feature 간 관계를 고려해 서비스로 그룹화하여 하나의 객체로 식별한다. 식별된 객체의 한 예인 '데이타 융합'이 그림 5에 나타나 있다. 그룹화시 고려해야 될 제약사항으로는 첫째, Serivce Fasade는 최상위 Feature를 선정하나 자식 Feature 간 분리되어야 하는 Feature는 자식 Feature를 Fasade로 선정하고 둘째, 계층 간 그룹화 과정에서 다른 Feature와 관계된 Feature가 속했을 경우에는 그룹화에서 제외하고 별도의 독립객체로 식별하며 셋째, 서비스 중복시 서비스 구조에 반드시 포함되어야 하는 Feature는 동일 Feature를 추가하여 재구성하는 것이다.

다음은 그룹화 과정을 통해 식별된 서비스와 Feature들을 후보 컴포넌트로 도출하는 과정이다. 본 논문에서는 객체리스트 6개 범주와 후보 컴포넌트 리스트 7개 범주를 선정하고, 그 상관관계를 표 2와 같이 정의하였다. 표에 나타난 C, R은 객체와 컴포넌트와의 친밀도를 나타내는 Close, Reference 측면의 값을 나타낸다.

이러한 상관관계를 그룹화 과정을 통해 식별된 객체 리스트와 후보 컴포넌트 리스트의 상호가중치를 다음 수식을 이용하여 계산한다. 여기서 M은 Mutuality로 객체와 컴포넌트와의 친밀도이다.

$$A(C)=5,\ A(R)=3$$

$$M(C_i,O_j)=\frac{A(C_i,O_j)}{\sum_{1\le k\le m} A(C_i,O_k)}$$

표 2 리스트 상관관계

리스트 상관관계		후보 컴포넌트 리스트(C)						
		Work Flow	Serv.	Intf.	Domain Tech.	Library	Comm.	Data Sharing
객체 리스트 (O)	Business	C	C			R		R
	Service		C	C			R	
	Interface		R	C			R	R
	Comput.		C	R	C	R		
	Connector		R	C			C	
	Data Entity	R				R		C

위 수식에서 $M(C_i, O_j)$는 후보 컴포넌트와 객체간의 친밀도 가중치 값이며, $A(C_i, O_j)$는 리스트 상관관계 표에서 i번째 후보컴포넌트와 j번째 객체의 친밀도 값으로 'C'는 5, 'R'은 3점을 부여해 전체 합과 해당 값을 나누어 가중치를 계산한다. 여기서 k는 1부터 m까지로 각 컴포넌트에 매칭되는 객체를 의미한다. 가중치 값을 계산 후 임계값을 정하여 클러스터링 과정을 통해 후보 컴포넌트를 식별하는 것이 최종 작업이 된다.

예 3) 후보 컴포넌트 도출과정을 통해 MSDFS의 후보 컴포넌트 리스트를 작성한다.

MSDFS에서 식별된 후보 컴포넌트에는 중앙제어(Domain Technology), 데이타 처리(Work Flow), 데이타 융합(Service), 다중 인터페이스(Interface) 등이다. 식별된 후보 컴포넌트의 일부분에 대해 각 후보 컴포넌트별 컴포넌트 범주, 관련 객체 그리고 객체의 범주가 표 3에 나타나 있다.

표 3 MSDFS의 후보 컴포넌트 리스트

후보 컴포넌트	범 주	관련 객체	범 주
중앙 제어	Domain Technology	DSP	Computation
		RT-OS	Computation
		Memory	Data Entity
		Power Cooling	Computation
		FPGA/ASIC	Computation
데이타 처리	Work Flow	A/D Conversion	Business
		D/A Conversion	Business
데이타 융합	Service	Data Control	Interface
		Correlation	Business
다중 Interface	Interface	센서 분류기	Service
		Access Manager	Connector
		Actuator	Business
		Distributer	Connector
...	...	...	...

3.3.2 하드웨어 종속성을 고려한 아키텍처 설계

시스템 최적화 설계를 위하여 임베디드 시스템 최적화 설계요건인 도출(extraction), 구성(configuration), 최적화(optimization) 프로세스[13]와 아키텍처 모델링 산출물인 객체 종속성 테이블, 서브시스템 모델, 배포 모델, 블록 다이어그램, 객체 모델을 매핑시켜 그림 7과 같은 하드웨어 종속성을 고려한 아키텍처 최적화 설계 프로세스를 제안한다.

아키텍처 모델링 과정에서 보여주는 뷰로는 '4+1뷰' 중 유스케이스뷰를 제외하고 논리뷰, 배포뷰, 프로세스뷰, 구현뷰 등 네가지이다. 여기서 유스케이스뷰는 도메인 분석단계에서 도출된 Feature 결정트리로 대체되기 때문에 제외한다. 아키텍처뷰를 이렇게 분리하는 것은

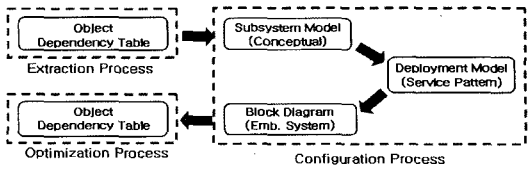

그림 7 아키텍처 최적화 설계프로세스

각 뷰마다 추구하는 특성이 다르기 때문에 분리된 뷰를 통해 아키텍트가 더욱 용이하게 제품계열 간의 협상을 분석할 수 있기 때문이다. 따라서 논리뷰는 서브시스템 모델, 배포뷰는 배포 모델, 프로세스뷰는 블록 다이어그램, 구현뷰는 객체 모델로 매핑하여 프로세스를 구성한다.

가. 도출프로세스 단계

도출프로세스 단계는 아키텍처 최적화를 위한 객체도출 단계로 서비스 구조를 통해 후보 컴포넌트 리스트를 작성한 결과를 통해 객체 종속성 테이블을 작성한다. 종속성 테이블은 임베디드 소프트웨어 특징인 하드웨어와의 종속성을 관련연구에서 검토한 것과 같이 대분류와 소분류로 구분하여 작성한다. 이렇게 작성된 객체 종속성 테이블은 다음 단계인 구성프로세스 블록 다이어그램의 입력값으로 들어간다.

나. 구성프로세스 단계

구성프로세스 단계에서는 서브시스템 모델, 배포 모델 그리고 최적화 구성의 핵심인 블록 다이어그램을 작성한다. 블록 다이어그램은 종속성 테이블을 통해 도출된 후보 컴포넌트를 통해 서브시스템 및 프로세스를 식별하고 이들을 분산시키기 위한 최적구조를 파악하여 이들 요소 모두를 표현하는 모델이다.

예 4) 아키텍처 최적화 설계프로세스의 도출프로세스 단계에서 MSDFS의 종속성을 후보 컴포넌트별로 판단하여 대분류와 소분류로 구분하여 MSDFS 종속성 테이블을 작성한다. 그런 다음 구성프로세스 단계에서의 블록 다이어그램을 작성하고 HP, OP, AP 구역을 표시한다.

MSDFS의 객체 종속성 테이블은 식별된 후보 컴포넌트를 주체로 대분류, 소분류를 검토하여 14개 객체를 대상으로 선정하였다. 중앙제어와 같은 하드웨어와 밀접한 관계를 갖고 운영체제 및 입출력 장치 간의 통신기능을 요구하는 부분은 HP로 구분하였고 소분류로서 ③과 ④로 분류하였다. 또한 데이타 처리와 같은 후보 컴포넌트는 OP와 AP 분류가 모호한 관계로 'OP or AP'로 구분하여 차후 블록 다이어그램을 통해 그 성격이 명확해질 경우 다시 재분류를 실시하였다. 이와 같이 하여 작성된 MSDFS의 종속성 테이블이 표 4에 나타나 있다.

다음으로 구성프로세스의 단계에서의 MSDFS 블록

표 4 MSDFS의 종속성 테이블

후보 컴포넌트	범 주	종속성 판단	
		대분류	소분류
중앙 제어	Domain Technology	HP	③ ④
데이타 처리	Work Flow	OP or AP	① ②
FPGA/ASIC	Computation	HP	⑤
다중 Interface	Interface	HP	④
Correlation	Service	AP	①
Tracker	Service	AP	①
Interpreter	Business	AP	①
Integration	Service	AP	①
Distribution	Service	HP or AP	① ④
DBMS	Data Entity	AP or HP	① ④
RT-OS	Computation	OP	③
Tran. Processor	Computation	OP	② ③
Memory	Data Entity	HP	⑤
...	...	...	...
다중 Interface	Interface	HP	② ⑤

① 기능을 구현한 애플리케이션 로직 자체
② 애플리케이션 기능수행 데이타 흐름제어 부분
③ OS가 애플리케이션의 기능 수행을 위해 하드웨어에 접근하는 부분
④ OS가 메모리, 프로세서, 입출력 장치 등의 하드웨어에 접근하는 부분
⑤ 하드웨어 자체기능(산술연산, ...)을 수행하는 부분

다이어그램을 작성하기 위해 중앙제어는 HP 구역에 배치하여 시스템 중앙에 놓고, HP & OP 구역에는 데이타 융합과 데이타 통합을 위치시켰다. AP로 분류된 센서 스트림과 다중 인터페이스의 경우에는 내부에 포함되는 프로세스가 OP 구역에 적용되는 부분이 있어 일부분은 HP & OP 구역에 배치하였다. 이와 같은 과정을 거쳐 작성된 MSDFS의 블록 다이어그램이 그림 8에 도시되어 있다. 이러한 배치전략은 하드웨어 및 객체간 이루어지는 커뮤니케이션을 최소화하고 공간배치상의 최적화를 달성할 수 있다. 또한 차후 시스템 변화성에 대한 하드웨어와 소프트웨어의 참조모델로서 핵심산출물 역할을 수행한다.

3.4 개발프로세스

본 논문에서 제안한 핵심자산 개발프로세스, 즉 효율적 요구사항 도출과 시스템 최적화 기법에 대한 전체 과정이 그림 9에 나타나 있다. 전체 과정을 살펴보면 먼저 도메인 모델링과 컴포넌트 모델링이라는 두 개의 단계(Phase)로 구분된다. 도메인 모델링 단계에는 스코핑, 비즈니스 모델링, Feature 모델링, 동적 모델링, 아키텍처 모델링의 5개 프로세스와 모델링 간에 도출되는 15개의 산출물을 정의하였다. 코드 개발단계인 컴포넌트 모델링에는 컴포넌트 모델링과 핵심자산 평가의 2개 프로세스와 4개 산출물을 정의하였다.

여기서 컴포넌트 모델링은 도메인 모델링에서 산출된

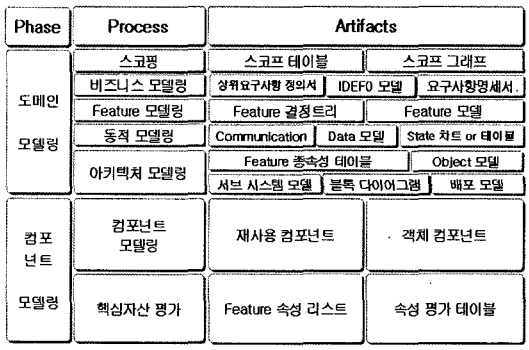

그림 9 핵심자산 개발프로세스 전체 과정

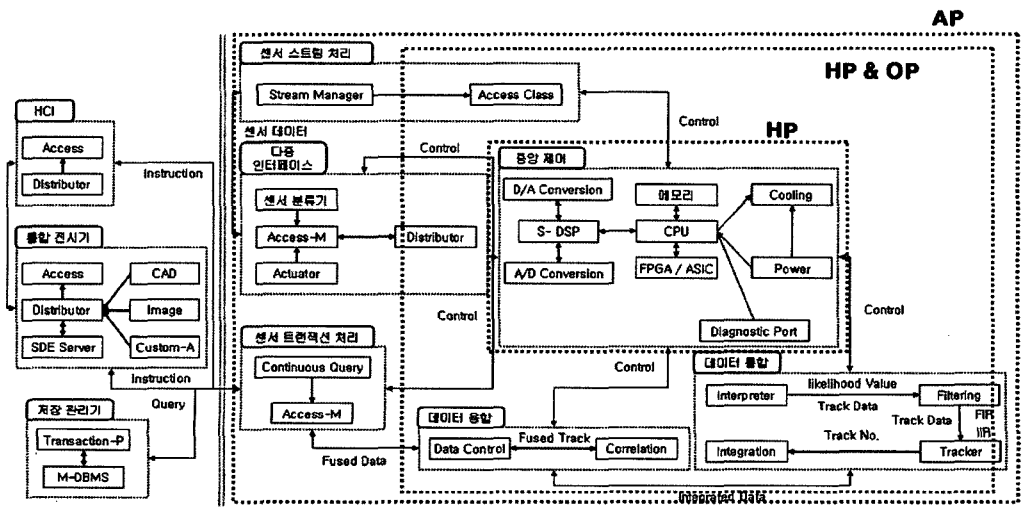

그림 8 MSDFS의 블록 다이어그램

후보 객체 및 컴포넌트 리스트를 통해 개발대상의 객체를 식별하여 이를 바탕으로 객체의 인터페이스와 구현 부분을 정의하는 객체 컴포넌트 개발단계와 이렇게 해서 개발된 객체 컴포넌트를 다양한 환경에 따라 적응이 가능하도록 만드는 재사용 컴포넌트 개발단계로 나뉘어 핵심자산을 도출해내는 제품계열공학의 영역공학에 마지막 단계로 정의된다.

4. 결 론

본 논문에서는 효율적인 임베디드 소프트웨어 개발을 위한 제품계열 중심의 핵심자산 개발프로세스 설계모델을 제안하였다. 제안한 모델은 효율적인 요구사항 도출을 위한 도메인 스코핑 단계에서부터 IDEF0 비즈니스 모델링, Feature 모델링, 동적 모델링, 하드웨어 종속성을고려한 아키텍처 모델링, 컴포넌트 모델링 단계까지 총 6단계로 구성하였다.

특히 본 논문에서는 서비스 구조를 통한 컴포넌트 도출을 위해 SOA에서 사용되는 전형적인 5단계 서비스 구조를 PLE를 적용한 4단계의 서비스 구조인 Service Fasade, Business Entity Layer, Process Layer, Data Layer로 재설정하였으며, 아키텍처 설계를 위해 논리뷰는 서브시스템 모델, 배포뷰는 배포 모델, 프로세스뷰는 블록 다이어그램, 구현뷰는 객체 모델로 매핑하여 프로세스를 구성하였다. 그리고 제안한 개발프로세스 설계모델이 어떻게 임베디드 소프트웨어 개발에 적용되는지를 보이기 위해 다중센서데이타 융합시스템에 적용한 결과를 설계과정마다 제시하였다.

참 고 문 헌

[1] Kyo C. Kang, Jae J. Lee, Donohoe, "Feature-oriented Product Line Engineering," IEEE Software, Vol. 19, pp.58-65, 2002.

[2] Klaus Schmid, Martin Verlage, "The Economic Impact of Product Line Adoption and Evolution," IEEE Software, Vol. 19, No. 4, pp.50-57, 2002.

[3] Carnegie Mellon Software Engineering Institute, A Framework for Software Product Line, Practice-Version 4.2, 2005.

[4] A. Sangiovanni-Vincentelli, G. Martin, "Platform-based Design and Software Design Methodology for Embedded Systems," IEEE Design & Test of Computers, Vol. 18, Issue 6, pp.23-33, 2001.

[5] M.A. Wehrmeister, L.B. Becker, F.R. Wagner, C.E. Pereira, "An Object-oriented Platform-based Design Process for Embedded Real-time Systems," 8th International Symposium on Object-Oriented Real-Time Distributed Computing, pp.125-128, 2005.

[6] J. Skene, W. Emmerich, "A Model-driven Approach to Non-functional Analysis of Software Architectures," 18th IEEE International Conference on Automated Software Engineering, pp.236-239. 2003.

[7] Hassn Gomaa, Designing Software Product Line with UML, Addison Wesley, 2005.

[8] Martin O.Hofmann, "Multi-Sensor Track Classification in Rotorcraft Pilot's Associate Data Fusion," Lockheed Martin Advanced Technology, 53rd AHSF, 1997.

[9] Liliana Dobrica, Eila Niemela, "Attribute based Product Line Development for Embedded Systems," VTT Publications, 2000.

[10] Costin Badica, "A new formal IDEF-based Modeling," First Balkan Conference in Informatic, BCI, 2003.

[11] S. Jones, "Toward An acceptable Definition of Service," IEEE Software, Vol. 22, Issue 3, pp. 87-93, 2005.

[12] Z. Stojanovic, A. Dahanayake, H. Sol, "Modeling and Design of Service-oriented Architecture," IEEE International Conference on Systems, Man and Cybernetics, Vol. 5, pp.4147-4152, 2004.

[13] John Linn, "Embedded Software Development Challenges in the DSP Era," System & Software Laboratory Texas Instruments Inc. Dallas Texas, Fifth International Symposium on Automous Decentralized Systems, 2001.

홍 기 삼

2000년 육군사관학교(공학사). 2005년 국방대학교(전산학석사). 관심분야는 임베디드 시스템 설계, 데이타베이스 최적화, 소프트웨어공학

윤 희 병

1983년 해군사관학교(이학사). 1986년 연세대학교(공학사). 1991년 미국 Naval Postgraduate School(전산공학 석사). 1998년 미국 Georgia Institute of Technology(전산공학 박사). 2002년 9월~현재 국방대학교 전산정보학과 부교수. 관심분야는 임베디드 소프트웨어 개발방법론, 임베디드 소프트웨어 아키텍처, 요구공학, 전술데이타링크, 공통운용환경