컴포넌트 기반 소프트웨어 개발을 위한 도메인 분석 및 설계 방법

(A Method of Domain Analysis and Design for Component-based Software Development)

하 현 주 † 문 미 경 †† 염 근 혁 †††

(Hyunju Ha) (Mikyung Moon) (Keunhyuk Yeom)

요 약 CBSD(Component-Based Software Development)는 이미 존재하는 소프트웨어 컴포넌트를 조립함으로써 시스템을 개발하는 방법이다. 컴포넌트를 이용하여 시스템을 개발하는 것은 개발시간과 비용을 줄이고, 생산성을 향상시키는 등 여러 가지 장점은 가진다. 그러나 여러 벤더에 의해 개발된 컴포넌트를 조립하는 것은 쉬운 일이 아니다. 이를 위해 컴포넌트가 어떤 문맥에서 사용되는 지 이해하는 것이 필요하며, 이 문맥은 아키텍쳐에 의해 결정된다. 따라서 컴포넌트 기반의 어플리케이션 개발은 아키텍쳐를 기반으로 하였을 때 가능하다. 또한 컴포넌트 기반 어플리케이션 개발을 위해서는 도메인 개념이 필수적이다. 컴포넌트의 재사용성을 높이기 위해서 관련된 시스템의 집합인 도메인에서 컴포넌트를 개발해야 한다.

본 논문에서는 컴포넌트 기반 어플리케이션 개발을 지원하는 도메인 아키텍쳐 개발방법을 제안한다. 도메인 아키텍쳐는 도메인 분석 및 설계 과정에서 생성되는 것으로 컴포넌트와 그들간의 관계를 표현한다. 도메인 아키텍쳐를 이용하여 컴포넌트를 개발함으로써 재사용성이 높은 컴포넌트를 효율적으로 개발할 수 있을 뿐만 아니라, 어플리케이션 개발 시 도메인에서 제공하는 정보를 이용하여 쉽게 어플리케이션을 개발할 수 있다.

Abstract Component-based software development(CBSD) is a method for building large software systems by integrating previously-existing software components. Software development method using components has several advantages such that reducing time to delivery and development costs, and increasing productivity, etc. But, integrating components developed from multiple vendors is too difficult. As a result, it is required the understanding on the context of use to develop reusable components. The context of use for software component is determined by software architecture. Therefore, it is possible to develop an application based on components if it is based on software architecture. Also, it is essential to consider domain concepts for CBSD. To increase the reusability of components, we should develop components in a domain which is a set of related systems.

In this paper, we proposed a domain architecture development methodology that supports component-based software development. Domain architecture that represents components and their relationship is produced through domain analysis and design process. We believe that component development methodology using proposed domain architecture can efficiently develop highly reusable components as well as easily develop an application using information acquired from domain architecture.

• 본 연구는 한국과학재단 목적기초연구(2000-2-30300-001-3)지원으로 수행되었음

† 정 회 원 : 삼성전자 정보통신총괄 통신연구소 연구원
sandsand@samsung.co.kr
†† 비 회 원 : 부산대학교 컴퓨터공학과
mkmoon@hyowon.cc.pusan.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
yeom@hyowon.cc pusan.ac.kr

논문접수 : 2000년 11월 8일
심사완료 : 2001년 8월 2일

1. 서 론

CBSD(Component-Based Software Development)는 이미 존재하는 소프트웨어 컴포넌트를 조립함으로써 시스템을 개발하는 방법이다[1]. 컴포넌트를 이용하여 시스템을 개발하는 것은 소프트웨어 개발 시간과 비용을 줄이고, 생산성을 향상시키며, 이미 검증받은 컴포넌트를 사용함으로써 위험성을 줄일 수 있다. 또한 컴포넌

트는 표준 아키텍처 상에서 동작하기 때문에 사용에 있어서 일관성을 높이며, 어플리케이션 개발을 위한 여러 가지 해결책을 제시하므로 가장 좋은 해결책을 선택할 수 있다는 장점을 가진다[2].

그러나 컴포넌트는 다른 아키텍처 스타일에서 설계되고, 다른 연결 표준(interconnection standard)상에서 구현되기 때문에 여러 벤더들로부터 개발된 컴포넌트를 조립하는 것은 쉬운 일이 아니다. 컴포넌트를 조립하여 어플리케이션을 개발하기 위해서는 컴포넌트가 어떤 문맥에서 사용될 것인가를 잘 이해하는 것이 필요하며, 이 문맥은 소프트웨어 아키텍처에 의해 결정된다. 따라서 컴포넌트를 기반으로 어플리케이션을 개발하는 것은 아키텍처를 기반으로 하였을 때만 가능하다[3].

또한 컴포넌트의 재사용성을 높이기 위해서는 관련된 시스템의 집합인 도메인에서 컴포넌트를 개발하여야 한다. 그리고 여러 시스템이 가지는 공통적인 특성을 포함하도록 컴포넌트를 추출하고, 각 시스템이 가진 다양한 특성을 컴포넌트에 적용할 수 있어야 한다.

본 논문에서는 아키텍처를 기반으로 하고, 도메인에서의 컴포넌트 개발을 지원하기 위하여 도메인 아키텍처 개발 방법을 제안한다. 도메인 아키텍처는 도메인 분석 및 설계 과정에서 생성되는 것으로, 여러 가지 관점에서 컴포넌트와 그들간의 관계를 표현한다. 따라서 컴포넌트가 어떤 환경에서 어떻게 동작할 것인가에 대한 충분한 정보를 제공하여 보다 쉽게 어플리케이션을 개발할 수 있다. 도메인 아키텍처를 기반으로 개발된 컴포넌트는 도메인에 속한 여러 시스템에서 사용될 수 있고, 각 시스템의 다양한 특성을 반영할 수 있다.

이 논문의 구성은 다음과 같다. 먼저 2장에서 관련 연구들을 설명하고, 3장에서 컴포넌트 기반의 도메인 분석과 설계 방법을 제안한다. 4장에서는 지능형 교통 시스템(Intelligent Transportation Systems : ITS)의 한 분야인 첨단 대중 교통 시스템(APTS : Advanced Public Transportation System)을 선택하여 제안한 도메인 분석 및 설계방법을 적용한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

2.1 컴포넌트 기반 소프트웨어 개발 방법

Catalysis는 UML을 기반으로 한 컴포넌트 기반 개발 방법을 지원한다[4]. 이 방법은 어떤 프로젝트에서도 적합하게 사용될 수 있도록 프로세스 패턴을 제공한다. 그림 1은 Catalysis 모델링의 3단계를 나타낸다.

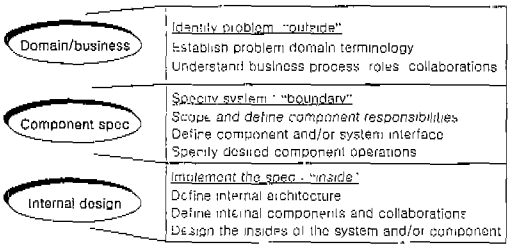

그림 1 Catalysis 모델링의 3단계

Domain/business단계에서는 고객이나 고객의 문제와 관련된 모든 개념을 인식한다. 즉, 개발하고자 하는 소프트웨어가 배치될 환경을 인식한다. Component spec 단계에서는 컴포넌트의 외부 행동을 정의한다. Internal design단계에서는 각 컴포넌트를 위한 인터페이스와 클래스를 설계한다.

Catalysis는 비즈니스 목적부터 프로그램 코드까지 이르는 컴포넌트 모델링 방법과 컴포넌트의 명확한 명세 기법을 제시하고 있다. 그러나 구체적으로 컴포넌트를 추출하고, 아키텍처를 구성하는 방법을 제시하기에는 미흡하다. 또한 Catalysis는 컴포넌트에 대한 가장 흥미있는 이론적인 접근 방법이나 사용하기에 너무 학술적이고 복잡하다[5].

Sterling에서는 Catalysis 방법을 사용하기 쉽도록 단순화하고 체계화하고, 확장했다[2][5]. Sterling의 CBSD 프로세스는 Requirements Definition, Analysis, Behavior Specification, Architecture 4단계로 이루어진다. Requirements Definition에서는 use case diagram을 사용하여 개발될 어플리케이션 요구사항을 정의하고, business type diagram을 사용하여 상위 레벨 비즈니스 개체와 그들간의 관계를 표현한다. Analysis에서는 어플리케이션의 문제 영역을 분석한다. Behavior Specification은 인터페이스를 인식하는 context modeling단계, 인식된 인터페이스를 상세히 분석하는 interface modeling단계와 인터페이스를 컴포넌트로 할당하고 컴포넌트를 기술하는 Component specification단계로 이루어져 있다. Architecture에서는 컴포넌트와 그들간의 상호작용을 표현한다.

Sterling에서는 COOL:Spex라는 case tool을 사용하여 컴포넌트 아키텍처 모델링과 컴포넌트 정의를 위한 인터페이스 기반 설계를 지원하고, COOL:Joe를 사용하여 분산환경에서 데이타베이스와 연동한 컴포넌트 모델링과 이에 대한 자바소스코드 생성을 지원한다.

2.2 도메인 엔지니어링(domain engineering)

도메인 엔지니어링의 목적은 소프트웨어 개발자들이 새로운 시스템을 개발하는 데 재사용 할 수 있는 부품들을 식별, 개발, 보급하는 것이다. 소프트웨어의 재사용은 특정 프로덕트 라인 내에서 계획되고 관리되어질 때 더 효율적이 된다. 프로덕트 라인 소프트웨어 개발은 그림 2에서 보는 바와 같이 2가지 생명공학 모델(2-life cycle model)로서 시각화 될 수 있다[6].

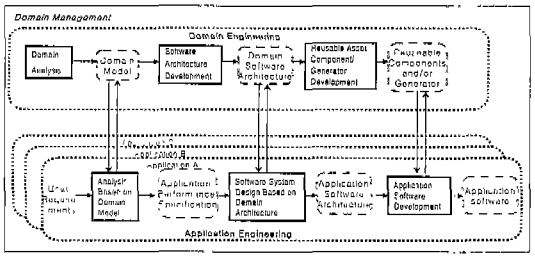

그림 2 프로덕트 라인 소프트웨어 개발

도메인 엔지니어링에서는 도메인 분석, 아키텍쳐 개발과 재사용 가능한 자산 개발을 포함하며 이들은 프로덕트 라인을 위한 모델, 아키텍쳐, 재사용 컴포넌트, 소프트웨어 생성기를 생산한다. 어플리케이션 엔지니어링은 이러한 자산들을 이용하여 프로덕트 라인 내에서 새로운 프로덕트를 만들어 내고, 프로덕트 라인의 자산에 의해 충족되지 않는 요구사항들은 도메인 엔지니어링에 피드백하여 프로덕트 라인에 결합되도록 한다.

이러한 도메인 엔지니어링을 대표하는 방법으로는 FODA (Feature-Oriented Domain Analysis)[7]와 ODM (Organization Domain Modeling)[9]이 있다.

FODA는 시스템의 집합 중에서 주도적인 또는 독특한 특성(feature)을 인식하는 것에 기초해 도메인을 분석하는 방법이다. 여기서는 '추상화(abstraction)'와 '정제화(refinement)' 두 가지 모델링 개념에 기반을 두고 있다. 이는 어플리케이션들 사이에 어떠한 차이도 존재하지 않는 수준까지 '추상화'되어야 한다는 것이고, 도메인 상에 있어서 특정 어플리케이션은 도메인 프로덕트들의 '정제화'로서 개발되어진다는 것을 의미한다. FODA는 Context analysis, Domain modeling과 Architecture modeling 과정으로 나뉜다. FODA를 소프트웨어 설계와 구현 단계 부분까지 확장시킨 모델로 FORM (Feature-Oriented Reuse Method)이 있다[8]. FORM는 재사용 가능한 아키텍처와 컴포넌트 개발, 그리고 도메인 엔지니어링에서 생성된 결과들을 이용한 어플리케이션 개발을 지원한다. 그러나 도메인에서 개발된 시스템이 이미 존재한다는 가정에서 출발하였고, 또한 UML을 기반으로 하지 않았다.

ODM(Organization Domain Modeling)은 형식적이고 반복적인 도메인 엔지니어링 방법을 제공한다[9][10]. 이 방법은 프로젝트와 도메인에 대한 관심을 두고 있고, 다양한 관점과 경험을 가지고 있으며, 다양한 용어를 사용하는 stakeholder에 초점을 두었다. 또한 도메인 예제들의 명시적인 집합인 exemplar를 기반으로 모델링하였다. ODM은 크게 Plan Domain단계, Model Domain단계와 Engineer Asset Base단계로 진행된다. Plan Domain단계에서는 도메인의 목적을 확립하고 범위를 한정하여 도메인을 정의한다. Model Domain단계에서는 도메인에 대한 정보를 수집하고, 도메인 모델을 개발한다. Engineer Asset Base단계에서는 아키텍쳐를 개발하고 구현한다.

ODM은 프로세스의 각 단계마다 입력되는 데이타, 수행되어야 할 활동, 결과 등을 구체적으로 기술하였다. 그러나 컴포넌트의 특성을 전혀 고려하지 않았고, UML을 기반으로도 하지 않았다. 또한 도메인 아키텍쳐 개발보다는 도메인 분석에 많은 비중을 두었다.

3. 도메인 분석 및 설계 방법

기존의 도메인 엔지니어링 방법들은 도메인을 분석하는 체계적인 방법들을 제공하고는 있으나, 컴포넌트를 인식하고 선택하고 조립하는 컴포넌트 기반 개발 프로세스에 필요한 정보들은 산출해 내지 못하고 있다. 따라서, 도메인 아키텍쳐를 개발하기 위한 도메인 분석 및 설계 방법을 제안하기에 앞서, 컴포넌트의 특성을 만족할 수 있도록 컴포넌트 기반 개발 프로세스를 정의한다.

3.1 CBSD 프로세스

CBSD는 이미 존재하는 소프트웨어 컴포넌트를 조립함으로써 시스템을 개발하는 방법이다[1]. CBSD는 프로덕트 라인 개발과 같이 크게 컴포넌트를 개발하는 도메인 엔지니어링과 컴포넌트를 조립하여 어플리케이션을 개발하는 어플리케이션 엔지니어링으로 나눌 수 있다. 본 논문에서는 컴포넌트 특성과 CBSD의 원칙을 고려하여 그림 3과 같은 CBSD 프로세스를 제안한다.

도메인 엔지니어링은 재사용 가능한 컴포넌트를 만드는 활동으로 도메인 모델과 도메인 아키텍쳐의 체계적인 개발을 지원한다[11]. 어플리케이션 엔지니어링은 고객의 요구사항을 분석하고 도메인에서 이미 개발된 컴포넌트들을 조립하여 어플리케이션을 개발하는 활동이다.

그림 3에서 제시한 CBSD 프로세스 중에서 컴포넌트 기반 어플리케이션 개발을 지원하는 도메인 아키텍쳐를

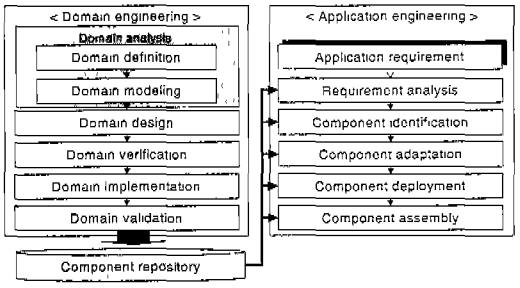

그림 3 CBSD 프로세스

개발하기 위한 도메인 분석 및 설계 방법을 개발한다. 도메인 분석은 도메인에서 해결해야 할 문제, 즉 도메인 요구사항을 인식하고 추출하는 과정으로 도메인 정의와 도메인 모델링으로 구성된다. 도메인 정의에서는 도메인의 목적과 범위를 정의하고, 그 과정에서 생성되는 도메인 명세서를 기술하는 방법을 개발한다. 도메인 모델링에서는 도메인 정의에서 생성된 도메인 명세서를 바탕으로 도메인 요구사항을 추출하고 모델링하는 방법을 정의한다. 도메인 설계에서는 도메인 요구사항을 해결하기 위해 컴포넌트를 추출하고, 그들로 구성된 도메인 아키텍처를 개발한다. 여기서 제안하는 각 단계별 산출물들-도메인 명세서, 도메인 모델, 그리고 도메인 아키텍처-은 표준 모델링 언어인 UML(Unified Modeling Language)을 기반으로 하여 표현된다.

3.2 도메인 정의(domain definition)

도메인 정의에서는 개발하고자 하는 도메인의 목적을 정의하고 범위를 한정하여 도메인 명세서를 생성한다. 명세서는 도메인에 대한 전반적인 정보를 제공하는 문서로, 도메인 요구사항을 도출하고, 아키텍처를 개발하기 위해 사용된다. 도메인 명세서는 그림 4와 같이 7가지 요소로 이루어진다.

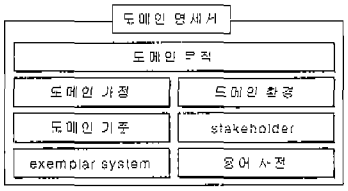

그림 4 도메인 명세서

3.2.1 도메인 목적

도메인 목적은 기능적인 측면을 중심으로 도메인의 대략적인 윤곽을 설명한다. 즉, 도메인 목적은 도메인의 기능적인 측면을 기술하기 위해 도메인에서 다루는 데이타의 특성, 그 데이타를 다루는 방법, 생성한 결과에 초점을 둔다. 따라서, 도메인 데이타, 도메인 오퍼레이션, 도메인 결과로 도메인 목적을 기술할 수 있다.

도메인 데이타는 도메인에서 주로 다루는 데이타로, 실세계에서 가지는 특성을 그대로 반영하도록 정의한다. 도메인 데이타는 표 1과 같은 항목으로 서술한다.

표 1 도메인 데이타의 서술 항목

kind	name	properties	subdata
· 상위 레벨 데이타 · is-a 관계	· 데이타 이름	· 데이타에 대한 설명 · 데이타의 제한조건 · 데이타가 가지는 속성	· 하위 레벨 데이타 · is-a 관계

표 1에서 name과 subdata의 관계는 subdata에 대한 서술에서 kind와 name의 관계가 된다.

도메인 오퍼레이션은 도메인 데이타 상에서 수행될 수 있는 오퍼레이션으로 표 2와 같이 5가지 항목으로 서술한다.

표 2 도메인 오퍼레이션의 서술항목

kind	name	needed data or device	properties	suboperation
· 상위 레벨 오퍼레이션 · is-a 관계	· 이름	· 수행을 위해 필요한 데이타 또는 장치	· 목적, 특징, 결과	· 하위 레벨 오퍼레이션 · is-a 관계

표 2에서 kind, name, suboperation의 관계는 도메인 데이타와 동일하다.

도메인 결과는 도메인 데이타에 도메인 오퍼레이션을 적용하여 생성한 결과로 표 3과 같은 항목으로 서술한다.

표 3 도메인 결과의 서술 항목

kind	name	target device	form	properties	subresult
· 결과의 상위레벨 · is-a 관계	· 이름	· 결과가 출력되는 장치	· 결과의 형태	· 결과에 대한 설명	· 결과의 하위 레벨 · is a 관계

표 3에서 kind, name, subresult의 관계는 도메인 데이타와 동일하다. Form은 image file, chart, table, text 등의 형태로 기술될 수 있다.

도메인 목적은 도메인 명세서의 다른 요소들을 이끌어내기 위한 토대로 사용된다. 그리고 어플리케이션 개발자에게 도메인에 대한 기본적인 정보를 제공한다.

3.2.2 도메인 가정

도메인 가정은 도메인에서 가정한 사항들을 표현한다. 도메인에서 제공하는 컴포넌트를 사용하기 위해 만족해야 할 전제조건들이 도메인 가정이 될 수 있다. 도메인에서 사용하는 데이타, 하드웨어 장치, 오퍼레이션, 결과, 환경 등에 대해 가정할 수 있으며, 도메인 가정은 도메인 기준, 환경 등을 정의할 때 사용된다.

3.2.3 도메인 기준

도메인 기준은 도메인의 경계를 명확하게 표현하기 위한 것으로, 도메인에 포함되는 어플리케이션인지 확인하는 수단이 되고 exemplar system의 선정 기준으로 사용된다.

도메인 기준은 도메인 데이타, 오퍼레이션, 결과의 형태로 기술되며, 포함기준(inclusion criteria)과 제외기준(exclusion criteria)으로 나눌 수 있다. 포함기준은 도메인에서 반드시 포함해야 할 요소이다. 제외기준은 도메인과 관련된 것이나 도메인에서 다루지 않는다고 도메인 가정에서 정의한 요소들로, 도메인 목적에서 정의한 도메인·데이타, 오퍼레이션 또는 결과와 관련이 있어야 한다. 또한 포함기준과 제외기준은 도메인 가정에서 데이타, 오퍼레이션, 결과에 대해 가정한 사항들을 만족할 수 있어야 한다.

3.2.4 exemplar system

Exemplar system은 도메인과 관련된 이미 존재하는 시스템으로 key-exemplar system과 counter-exemplar system으로 나눈다. Key-exemplar system은 개발하고자 하는 도메인에 완전히 포함될 수 있는 시스템으로, key-exemplar system에서 공통성과 다양성을 추출한다. Counter-exemplar system은 도메인의 경계 밖에 존재하는 시스템으로, 도메인과 관련된 것이나 도메인에서 다루지 않는다고 가정한 요소들을 가진 시스템이다. Counter-exemplar system은 도메인 환경의 외부요소들을 추출하기 위해 사용한다. 그림 5는 도메인 기준과 exemplar system과의 관계를 나타낸다.

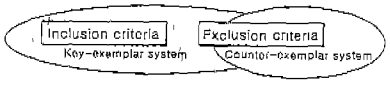

그림 5 도메인 기준과 exemplar system과의 관계

3.2.5 도메인 환경

도메인 환경은 도메인 외부 환경과 도메인 내부 환경으로 나눈다. 도메인 외부 환경은 도메인과 도메인 외부 요소들과의 상호작용을 나타냄으로써 도메인의 경계를 명확하게 표현한다. 도메인 외부 환경은 collaboration diagram을 사용하여 그림 6과 같이 표현한다.

그림 6에서 vertical component는 도메인의 기능 수행을 위하여 필요한 컴포넌트이나 다른 도메인에서 이미 정의하여 개발된 컴포넌트이다. Horizontal component도 vertical component와 같이 도메인의 기능 수행을 위하여 필요한 컴포넌트이나, 특정 도메인에 포함된 것이 아니라 도메인과 상관없이 사용될 수 있는 컴포넌트이다. 3.2.4에서 설명한 counter-exemplar system의 컴포넌트가 vertical component나 horizontal component가 된다. 그 외 도메인에서 사용하는 DB나 H/W device 등이 도메인 외부 요소가 될 수 있다.

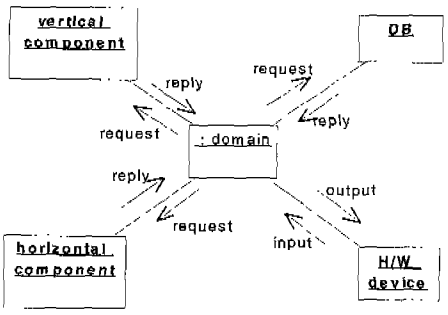

그림 6 도메인 외부 환경

그림 6에서 보듯이 도메인과 도메인 외부요소와의 상호작용은 서로 간에 교환되는 데이타 또는 메시지로 표현한다.

도메인 내부 환경은 도메인 내에서 반드시 분산되어야 하는 요소들과 그 요소가 제공하는 기능을 collaboration diagram을 사용하여 표현한다. 도메인 내부 환경은 도메인이 단일화된 시스템(monolithic system)인 경우 필요하지 않고, 도메인이 클라이언트-서버 환경이나 분산환경에서 동작한다고 가정한 경우에 서술한다. Collaboration diagram에서 표현된 정보는 컴포넌트 추출단계에서 사용된다.

3.2.6 domain stakeholder

Domain stakeholder는 도메인에서 정의한 활동에 참여하거나 도메인에서 제공하는 서비스에 관심있는 사람들로, 도메인은 이들의 요구사항을 수용할 수 있어야 한다. 그림 7은 domain stakeholder의 추출기준을 제시한다.

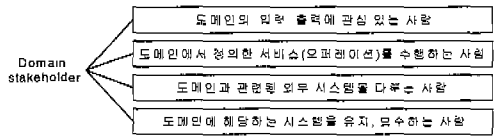

그림 7 domain stakeholder 추출 기준

3.2.7 도메인 사전

알파벳 순서로 도메인 명세서에서 사용한 용어의 뜻을 정의한다.

3.3 도메인 모델링 (domain modeling)

도메인 모델링은 도메인 명세서를 바탕으로 공통성과 다양성을 추출하며, 요구사항을 인식하고 분석하여 표현한다. 그 결과 도메인 모델을 생성한다. 도메인 모델은 도메인의 요구사항을 표현하는 도메인 요구사항 모델, 그 요구 사항에 대한 서술서, 도메인에서 저장해야 할 데이타를 표현하는 도메인 개념 모델로 구성된다.

3.3.1 공통성과 다양성 추출

exemplar system으로부터 동일한 목적을 가지는 기능, 즉 동일한 종류의 데이타를 가지고 동일한 종류의 결과를 생성할 때 그 기능을 공통성으로 추출한다. 다양성은 추출된 공통성을 기반으로 동일한 종류의 데이타이나 그 데이타의 특성이나 범위가 다를 때, 기능을 수행하는 방법이나 수단이 다를 때, 결과의 형태가 다를 때 다양성으로 정의한다. 그 외에도 특정 기능에 대해 그 기능을 수행하기 위한 하드웨어 장치의 역할을 기준으로 각 장치의 특성, 그리고 설계 또는 구현단계에서 고려해야 할 품질, 운용 환경 등이 다양성이 될 수 있다.

추출된 다양성은 요구사항 서술서에 반영되어, 컴포넌트 추출 후 컴포넌트 다양성으로 정의된다.

3.3.2 도메인 요구사항 모델

도메인 요구사항 모델은 도메인에서 추출된 요구사항들을 표현한다. 도메인 요구사항 모델을 생성하기 위하여 먼저 도메인의 요구사항을 추출하여야 한다. 도메인 요구사항은 그림 8과 같은 항목에서 추출한다.

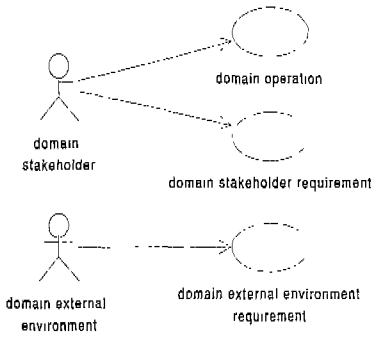

그림 8 도메인 요구사항 추출

먼저 domain stakeholder와 도메인 외부 환경으로부터 actor를 추출한다. 두 번째, 이러한 actor가 가지는 요구사항, 그리고 도메인 오퍼레이션을 요구사항으로 추출한다. 마지막으로 actor에게 결과를 제공하는 단위로 요구사항을 재정의하여 use-case diagram을 작성한다. 추출된 요구사항들은 use, include, extend 3가지 관계를 가질 수 있다[12]. 이러한 use case 간의 관계는 컴포넌트를 추출하는 기준으로 사용된다.

3.3.3 도메인 요구사항 서술서

도메인 요구사항 서술서는 도메인 요구사항 모델에서 정의한 각 요구사항에 대한 정보를 제공한다. 일반적으로 use-case diagram에서 use case description은 use case의 목적, 초기화 방법, actor와 use case 간에 교환되는 메시지의 흐름, use case의 종료 방법의 요소로 구성된다[12][13]. 본 논문에서는 각 도메인 요구사항에 대해 텍스트로 그림 9와 같은 항목을 기술한다.

1) goal
2) *precondition*
3) flow of events
   ① main flow
   - *describe what systems does and that the actor does*
   - how use case start
   - system interaction with the actors and what they exchange
   - paths of execution that are not allowed
   - usage of objects, values, and resources in the system
   - how and when the use case ends
   ② alternative flow
4) postcondition
   ① successive postcondition
   ② failed postcondition
5) variation

그림 9 도메인 요구사항 서술서

Goal은 use case가 달성하고자 하는 궁극적인 목적이 무엇인지 서술한다. Precondition은 어떤 상황에서 use case를 실행할 수 있는 지 서술한다. *Flow of events*는 use case와 actor간에 정보를 인지하고 수정하고 검색하기 위해 교환되어야 할 메시지 또는 이벤트의 흐름을 나타내고, main flow와 alternative flow로 나눈다. Main flow는 가장 공통적으로 수행될 수 있고, actor에게 가장 명백한 결과를 생성할 수 있는 흐름을 나타낸다. Alternative flow는 use case가 어떤 상황이나 예외에 따라 수행될 수 있는 대안을 말하며 메시지 흐름의 다양성을 표현한다. Postcondition은 use case가 성

공적으로 수행되었을 때 시스템의 상태와 결과를 서술하는 Successive postconditon과 use case가 성공적으로 수행되지 못했을 때 상태와 결과를 서술하는 failed postcondition으로 나눌 수 있다. Variation은 exemplar system에서 추출한 다양성을 서술한다.

3.3.4 도메인 개념 모델

도메인 개념 모델은 요구사항을 만족하기 위해서 도메인에서 저장되어야 할 데이타와 그들간의 관계, 데이타의 속성을 표현한다. 도메인 개념 모델에서 기술되어야 하는 데이타는 도메인 데이타와 요구사항 서술서에서 추출된다. 도메인 요구사항 서술서의 flow of events 중 시스템에서 사용하는 객체, 데이타, 자원에 대한 서술에서 저장되어야 하는 데이타들을 추출한다. 그리고 추출된 데이타와 일치하는 도메인 데이타(표 1)의 정보를 이용하여 데이타를 구체적으로 class diagram을 사용하여 표현한다.

3.4 도메인 설계 (domain design)

도메인 설계에서는 도메인을 구성하는 컴포넌트와 그들간의 관계를 인식하고 표현하여 도메인 아키텍처를 생성한다. 이러한 도메인 아키텍처는 어플리케이션 개발 시, 요구사항을 만족하는 컴포넌트를 인식하고 어플리케이션 특성에 맞는 컴포넌트를 선택하고 실행 시 컴포넌트간의 관계를 고려하여 프로세스에 컴포넌트를 할당하고, 컴포넌트의 관계를 고려하여 컴포넌트를 조립하기 위한 기반이 된다. 따라서 본 논문에서 어플리케이션 개발 단계에서 필요한 정보들을 바탕으로 도메인 아키텍처를 여러 뷰로 나눈다. 이로 인해 다양한 stakeholder의 관심을 분리하여 설명할 수 있고 기능적인 요구사항과 비기능적인 요구사항을 분리하여 표현할 수 있다[14].

표 4 어플리케이션 개발 단계와 필요한 정보

	수행하는 일	필요한 정보
requirements analysis	· 요구사항 분석 · 해당 도메인 찾기 · 도메인 모델을 참조, 요구사항 재정의	· 도메인 명세서 · 도메인 모델
component identification	· 요구사항을 만족하는 컴포넌트 인식	· 특정 요구사항을 만족하는 컴포넌트
component adaptation	· 컴포넌트 커스터마이즈 또는 · 어플리케이션에 맞는 컴포넌트 선택	· 커스터마이즈 지점 · 컴포넌트 선택 방법
component deployment	· 컴포넌트를 프로세스에 할당	· 실행 시 컴포넌트간의 관계
component assembly	· 커넥터 선택 · 컴포넌트 조립	· 컴포넌트간의 연결 관계 · 커넥터 정보

표 4는 어플리케이션 개발 단계와 각 단계마다 수행되는 일, 필요한 정보들을 나타낸다.

표 4를 바탕으로 그림 10과 같은 도메인 아키텍처 뷰를 제안한다.

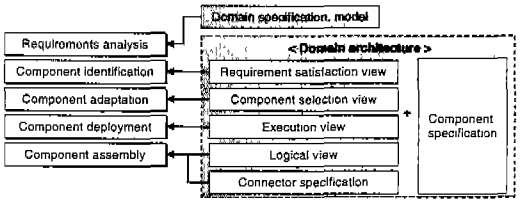

그림 10 도메인 아키텍처 뷰

도메인 아키텍처 뷰는 4가지 형태의 뷰들과 컴포넌트와 컨넥터에 대한 명확한 명세를 위해 2가지의 specification으로 표현된다. 그림 11에서는 각 뷰 간에 어떤 정보를 주고받는 지 보여준다.

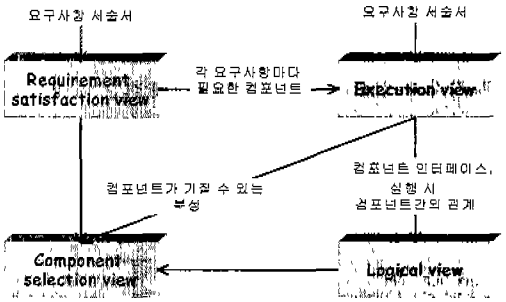

그림 11 도메인 아키텍처 뷰간의 관계

3.4.1 Requirement satisfaction view

Requirement satisfaction view는 특정 요구사항을 수행하기 위해 필요한 컴포넌트들을 표현한다. 따라서 각 요구사항마다 뷰가 생성된다.

각 요구사항마다 필요한 컴포넌트들을 표현하기 위해서 요구사항 단위로 컴포넌트를 추출한다. 그 과정은 우선 도메인 요구사항 모델에서 제시하는 use case 간의 관계를 고려하여 관련된 use case를 하나로 통합하는 것이다. 통합이 끝나면, 4-tier 아키텍처를 기반으로 하여 요구사항 서술서에서 인터페이스, 컨테이너, 컨트롤 및 프로세싱, 데이타 컴포넌트를 추출한다. 검출된 컴포넌트들은 도메인 개념 모델을 바탕으로 각 컴포넌트가 동일한 데이타 상에서 동작하고, 요구사항을 만족하기 위하여 서로간의 통신이 필요하다면 하나의 컴포넌트로 통합

한다. 이러한 과정을 거쳐 Requirement satisfaction view는 use-case diagram을 사용하여 그림 12와 같이 표현한다.

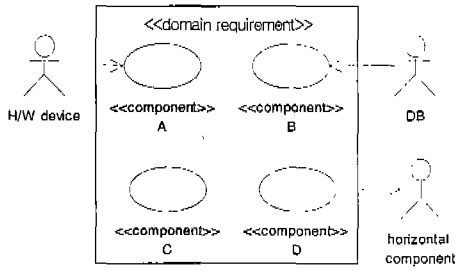

그림 12 requirement satisfaction view

3.4.2 Component selection view

Component selection view는 동일한 인터페이스와 기능을 제공하는 컴포넌트들이 가질 수 있는 여러 특성에 대해 표현한다. 이를 이용하여 어플리케이션 개발 시 동일한 명세를 가진 컴포넌트 중에서 어플리케이션의 목적에 맞는 컴포넌트를 선택할 수 있다. 설계 또는 구현된 각각의 컴포넌트가 가지는 비기능적인 특성, 컴포넌트 아키텍쳐, 컴포넌트 버전, 구현 기술, 알고리즘 등에 대해 표 5와 같이 테이블을 사용하여 표현한다.

표 5 component selection view

component ID	attribute_1	...	attribute_N
ID_value			
...			
..			

3.4.3 Execution view

Execution view는 특정 요구사항을 수행하기 위해 발생하는 컴포넌트간의 상호작용을 표현한다. Execution view는 성능이나 통신, 동기화와 관련된 문제를 다룸으로써 어플리케이션 개발 시 컴포넌트를 프로세스에 할당할 때 필요한 정보를 제공하고, 컴포넌트 인터페이스를 추출하기 위해 사용될 수 있다. 또한 두 컴포넌트간의 실행 시 관계를 나타냄으로써 커넥터에 대한 정보를 제공한다.

Execution view는 collaboration diagram을 사용하여 표현한다. 컴포넌트간에 교환되는 각 메시지에는 collaboration number를 부여하여 동시에 실행될 수 있는 인터페이스를 표현한다. 또한 메시지 분석을 통해 컴포넌트의 인터페이스를 추출한다. 그림 13은 execution view를 나타낸다.

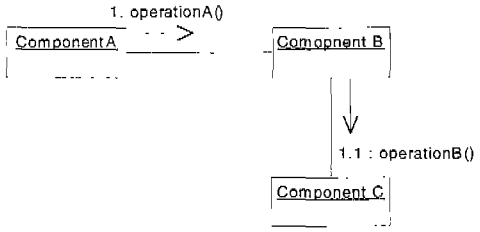

그림 13 execution view

3.4.4 Logical view

Logical view는 도메인 전체에 대해 컴포넌트와 커넥터, 그들간의 관계를 class diagram을 사용하여 표현한다. Logical view는 요구사항마다, 또는 컴포넌트 단위마다 생성된 다른 뷰들을 하나로 통합시켜주는 역할을 한다.

Logical view의 컴포넌트는 UML의 package notation을 사용하여 표현한다. Package는 private, protected, public, implementation visibility를 가질 수 있고 자신의 행동을 알리기 위해 인터페이스를 가진다. 일반적인 package와 컴포넌트를 구별하기 위해 <<component>>라는 stereotype를 사용한다.

Logical view의 커넥터는 execution view로부터 추출한다. Execution view에서 제공하는 각 컴포넌트간에 실행 시 발생하는 관계를 기준으로 어떤 특성을 가지는 커넥터가 필요한 지 인식한다. 커넥터는 interface notation을 사용하여, attribute 부분에는 커넥터가 연결시키는 컴포넌트 이름을, operation 부분에는 커넥터의 연결방법을 기술한다.

그림 14는 class diagram을 사용하여 logical view를 표현한 것이다.

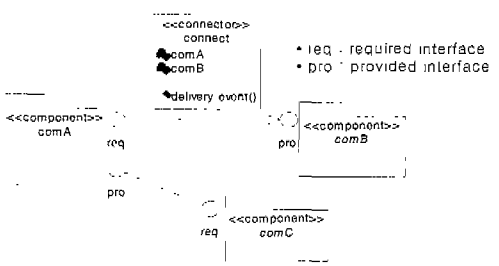

그림 14 logical view

다른 컴포넌트에게 서비스를 요청하는 required interface는 서비스를 제공하는 provided interface와 연결된다. 두 컴포넌트 인터페이스의 연결관계가 일반적인 function call이 아닌 경우, connector를 사용하여 둘 사이의 연결관계를 명확하게 표현한다.

3.4.5 Component specification

Component specification은 각 컴포넌트가 무엇을 하는지, 그리고 그 컴포넌트를 어떻게 사용할 수 있는 지에 대한 정확한 정보를 제공한다. 컴포넌트의 정확한 능력을 이해하기 위해서는 인터페이스 요소의 의미, 그들의 관계, 가정한 동작 환경, 품질 속성과 같은 점을 서술해야 한다[15]. 본 논문에서는 정확하고 검증할 수 있는 컴포넌트에 대한 정보를 제공하기 위해 component specification을 정의한다.

Component specification은 컴포넌트의 목적, 컴포넌트 행동에 대한 서술, 컴포넌트 인터페이스 서술, 동일하거나 유사한 기능을 제공하는 컴포넌트, 다른 컴포넌트와의 관계, 비기능적인 특징 등으로 구성된다. 그림 15는 component specification을 구성하는 요소를 보여준다.

Purpose는 컴포넌트의 일반적인 목적을 정의한다. Configuration interface는 컴포넌트가 가지는 유일한 속성 정보들로, 개발된 컴포넌트를 검색하기 위한 수단으로 사용된다. Configuration interface를 통해 컴포넌트 식별자, 버전, 개발 날짜, 제조업체 정보를 알 수 있으며, 이들 정보는 변경 불가능하다. Related component 부분에서는 정의하고 있는 컴포넌트와 동일하거나 유사한 문제를 해결하는 다른 컴포넌트를 서술한다. Port는 컴포넌트가 가지는 인터페이스 중에서 외부에서 볼 수 있는 오퍼레이션을 말한다. Port에서는 컴포넌트의 인터페이스에 정의된 각 오퍼레이션에 관한 정보를 제공한다. 각 오퍼레이션에 포함되어야 할 정보에는 intent, operation signature, precondition, postcondition, component dependency, customization point, quality가 있다.

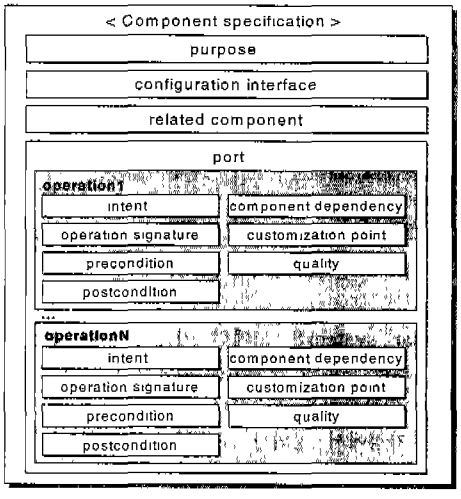

그림 15 component specification

Component specification은 좀 더 정확한 정보 제공을 위하여 OCL(Object Constraint Language)를 사용하여 표현한다[16][17]. 또한 본 논문에서는 각 컴포넌트에 대하여 state diagram을 작성하여 컴포넌트의 행동에 대해서도 명확하게 표현한다.

3.4.6 Connector specification

Connector specification은 컴포넌트와 컴포넌트를 연결시켜주는 커넥터에 대해 커넥터 타입, 커넥터가 연결시켜주는 컴포넌트, 연결되는 방법 등의 요소로 서술한다[18]. 그림 16은 connector specification을 이루는 요소를 나타낸다.

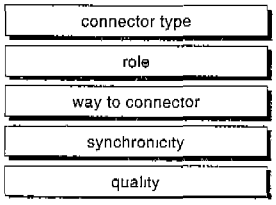

그림 16 connector specification

Connector type은 이벤트, 메시지, 큐와 같은 커넥터의 종류를 말한다. Role은 커넥터가 연결시켜주는 컴포넌트의 타입, way to connector는 컴포넌트가 연결되는 방법을 말한다. Synchronicity는 커넥터가 컴포넌트간의 동시성을 지원해주는지, quality는 커넥터가 가지는 성능, 신뢰성과 같은 특성을 말한다. Connector specification도 좀 더 정확한 정보 제공을 위하여 OCL을 사용하여 기술한다.

4. 사례연구

본 논문에서는 기존의 교통시스템에 전자, 통신, 제어, 컴퓨터 등 첨단 기술을 접목시켜 신속, 저렴하고 안전한 교통환경을 확보하며 운영의 효율화를 기한 지능형 교통 시스템(Intelligent Transportation Systems : ITS)[19][20]의 한 분야인 첨단 대중 교통 시스템(APTS : Advanced Public Transportation System)을 선택하여 제안한 도메인 분석 및 설계방법을 적용하였다. APT에서 제공하는 서비스는 그림 17과 같다[21].

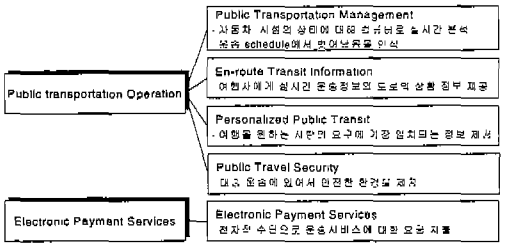

그림 17 APTS에서 제공되는 서비스

4.1 APTS 도메인 명세서

그림 17을 참고로 하여 개발하고자 하는 APTS 도메인에 대하여 그림 18과 같이 도메인 데이타, 도메인 오퍼레이션, 도메인 결과로 이루어진 도메인 목적을 정의한다.

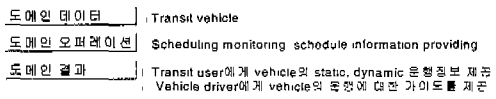

그림 18 APTS의 도메인 목적

3.2.1에서 제시한 방법을 적용해서 APTS 도메인 데이타, 도메인 오퍼레이션, 도메인 결과 각각을 구체적으로 서술하면 표 6, 7, 8과 같다.

표 6 도메인 데이타의 구체화

kind	name	properties	subdata
transit vehicle	bus	도로상에서 운행(routing 정보 필요)	city bus express bus
	train	rail상에서 운행(정해진 경로를 따라 운행) type : 새마을, 무궁화, 통일 class : 침대, 특실, 일반	
	subway	rail 상에서 운행(정해진 경로를 따라 운행) 노선에 따라 그룹번호 가진다.	

표 7 도메인 오퍼레이션의 구체화

kind	name	needed data or device	properties
schedule monitoring	check vehicle schedule	vehicle number location from vehicle location tracking system	vehicle이 스케줄대로 운행되고 있는 지 확인
schedule information providing	search route for transit user	origin, destination vehicle type	목적지까지 가는 vehicle group에 대한정보와 요금, 걸리는 시간 등을 알려줌
	provide vehicle group's schedule info	vehicle group number	특정 vehicle group의 운행표 제공
	search vehicle group by access point	access point	특정 access point를 경유하는 vehicle group에 대한 정보 제공

표 8 도메인 결과의 구체화

kind	name	target device	form	properties
static schedule info	vehicle route	display terminal	map, text	특정 vehicle의 group의 노선 및 배차간격, 출발시간
	route for transit user	display terminal	map, text	사용자가 원하는 목적지까지의 경로 및 vehicle에 대한 정보(group number, fare, running time)
	vehicle group at access point	display terminal	text	특정 access point를 경유하는 vehicle group과 노선정보
dynamic schedule info	expected arrival time at access point	display terminal	text	vehicle의 도착예정 시간
	delay	display terminal	text	delay 정도, 이유
running guide	schedule deviation	MDT	text	예정된 스케줄과의 차이
	new route	MDT	map	목적지까지 빠른 시간 내에 가기 위한 새로운 경로

제시한 도메인의 목적 하에서, APTS 도메인을 어떠한 전제조건을 가지고 개발할 것인지 그림 19와 같이 도메인 가정을 서술한다.

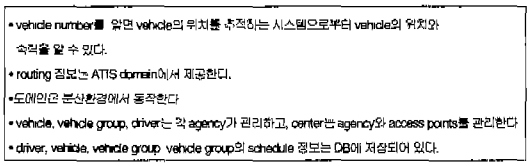

그림 19 APTS 도메인 가정

도메인의 목적을 만족하면서 도메인에서 반드시 다루어야 할 포함기준과 도메인과 관련되었으나 다루지 않을 제외기준을 정의하면 그림 20과 같다.

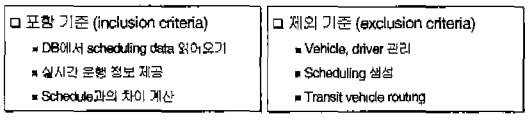

그림 20 APTS 도메인 기준

Exemplar system은 도메인 기준에 따라 선정한다. 그리고 정의한 도메인 명세서를 이용하여 그림 21과 같이 도메인 외부환경을 기술한다.

도메인 가정에서 정의한 도메인이 동작할 환경이나, 도메인의 주요 기능에 따라 그림 22와 같이 도메인 내부환경을 기술한다.

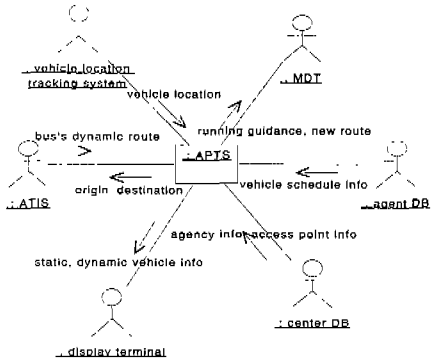

그림 22 도메인 외부 환경

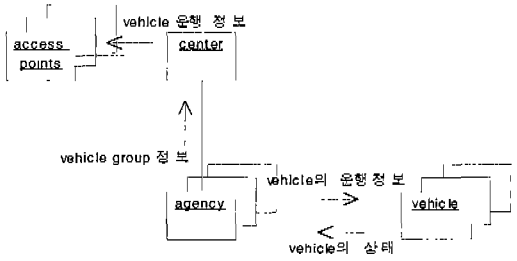

그림 21 도메인 내부 환경

Domain stakeholder는 도메인 환경과 도메인 오퍼레이션, 결과로부터 'transit user'와 'vehicle driver'를 추출한다(표 9).

지금까지 도메인 명세서에서 사용한 용어들을 정의한 도메인 사전을 생성한다.

4.2 APTS 도메인 모델

도메인 요구사항 모델을 작성하기 위하여 우선 domain stakeholder의 요구사항을 표 9와 같이 추출한다.

Domain stakeholder 그리고 도메인 외부 환경을 actor로 추출하고, 각 actor에 대한 요구사항을 추출하여 그림 23과 같이 요구사항 모델을 작성한다.

표 9 domain stakeholder 요구사항 추출

추출기준	name	역할 또는 요구사항
도메인 결과	transit user	· vehicle의 운행정보 요구 · 목적지까지 가기 위한 최단 시간 경로 요청 · 특정access point를 경유하는 vehicle 정보 요청
도메인 결과, 외부 시스템 다름	vehicle driver	· 운행중인 vehicle에 대한 상태 확인 요구 · AVL(bus를 위한 vehicle tracking system)를 활성화

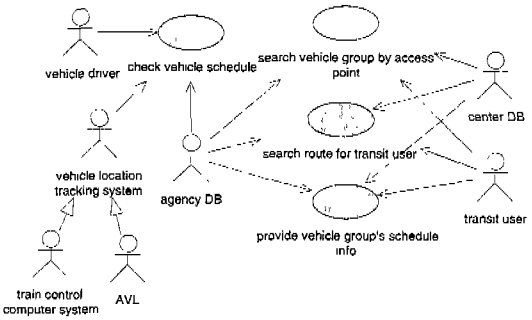

그림 23 APTS 도메인 요구사항 모델

그림 23에서 제시한 4가지 요구사항에 대해 각각 요구사항 서술서를 작성한다. 그림 24는 vehicle의 현재위치에 대한 schedule과의 차이를 계산하는 'check vehicle schedule'에 대한 서술서를 나타낸다. Main flow를 살펴보면 'vehicle schedule' 정보가 저장되어야 함을 알 수 있다. 저장되어야 할 'vehicle schedule' 정보는 'vehicle ID', 'access point'의 방문 순서(vehicle의 운행경로), 도착시간 등이다. 또한 구체화된 도메인 데이타(표 6)로부터 vehicle은 차량 번호, 소유자, 운전자, 좌석 수 등에 대한 정보를 저장해야 함을 알 수 있다. 이와 같이 요구사항 서술서와 도메인 데이타로부터 인식한 도메인 개념 모델을 그림 25와 같이 나타낸다.

1) goal
- 운행중인 vehicle이 schedule 대로 운행되고 있는 지 확인한다

2) precondition
- vehicle은 운행 중에 있다.
- AVL은 activate 되어 있다

3) flow of events

- Main flow

① vehicle driver가 vehicle의 운행상태를 확인해줄 것을 요구한다
② vehicle system은 vehicle location tracking system로부터 vehicle의 현재위치와 속도를 입력받는다
③ agency system에 vehicle group ID, vehicle ID vehicle 현재위치 속도를 넘겨준다.
④ agency system로 해당 vehicle의 scheduling 정보를 읽어온다
(vehicle group의 schedule 표에 포함된 특정 vehicle의 schedule 정보)
⑤ 현재위치를 기준으로 각 access point에 vehicle의 도착예정 시간, delay 시간을 계산한다.
⑥ vehicle이 예정보다 늦다면 ATIS domain에 vehicle의 새로운 경로를 요구한다
⑦ vehicle driver와 center에 delay 정보를 알려주고 vehicle driver에게는 새로운 경로도 알려준다
⑧ center는 agency에서 넘겨받은 정보를 해당 access point로 보낸다
⑨ access point는 center로부터 받은 정보를 display한다

- Alternative flow
④에서 vehicle의 예정보다 늦지않다면 ATIS에 새로운 경로를 요구하지 않는다
⑦ 에서 vehicle이 예정보다 빨리 운행되고 있다면 vehicle driver에게 얼마나 빨리지에 대해 알려준다
⑦ 에서 vehicle이 예정대로 운행되고 있다면 vehicle driver에게 앞으로의 schedule 정보를 알려준다

4) postcondition
- Successive postcondition
  - Use case는 vehicle driver에게 vehicle의 운행상태에 대한 정보를 제공하고 종료한다
  - Vehicle driver는 schedule 상에서의 delay에 대한 정보와 새로운 경로를 제공받는다.

5) variability
- bus일 경우에는 bus 내부에 있는 AVL로부터 vehicle의 현재위치를 입력받고
- train, subway일 경우에는 rail상의 train control computer system으로부터 위치를 입력받는다

그림 24 check vehicle schedule 요구사항 서술서

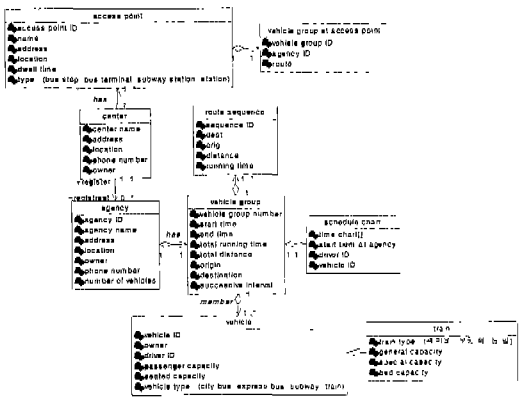

그림 25 APTS 도메인 개념 모델

4.3 APTS 도메인 아키텍쳐

도메인 아키텍쳐를 구성하기 위하여 우선 컴포넌트를 추출하여야 한다. 컴포넌트는 요구사항 서술서를 중심으로 도메인 개념 모델, 도메인 내부 환경에서 제공하는 정보를 이용하여 추출한다. 이러한 과정을 거쳐 네 개의 도메인 요구사항에 대해 추출한 컴포넌트들은 그림 26과 같다.

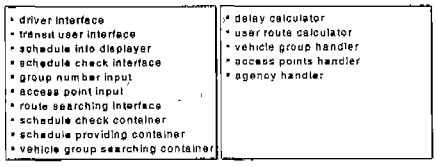

그림 26 APTS 컴포넌트들

컴포넌트를 추출하고 난 후, 각 요구사항마다 어떠한 컴포넌트가 필요한지 그림 27과 같은 requirement satisfaction view를 작성한다.

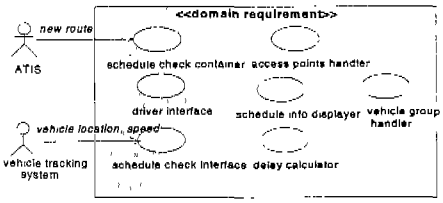

그림 27 check vehicle schedule의 requirement satisfaction view

Execution view는 각 요구사항에 대해 실행 시 발생하는 컴포넌트간의 상호작용을 collaboration diagram을 사용하여 그림 28과 같이 작성한다.

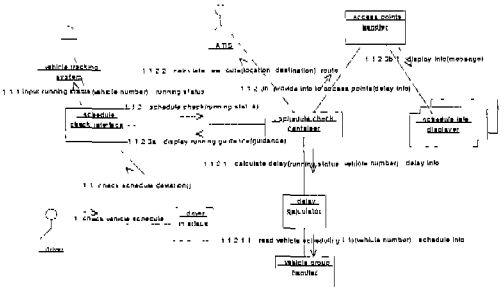

그림 28 check vehicle schedule의 execution view

'schedule check container'에서 호출하는 '1.1.2.3a : display running guidance(guidance)'와 '1.1.2.3b : provide info to access points(delay info)'는 collaboration number를 통해 동시에 실행될 수 있음을 알 수 있다.

Execution view에서 각 컴포넌트간의 관계를 고려하여 필요한 커넥터를 그림 29와 같이 추출하고, 도메인에 대해 logical view를 작성한다.

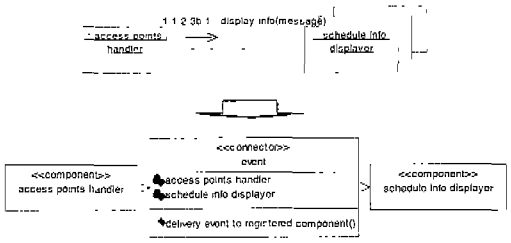

그림 29 커넥터 추출

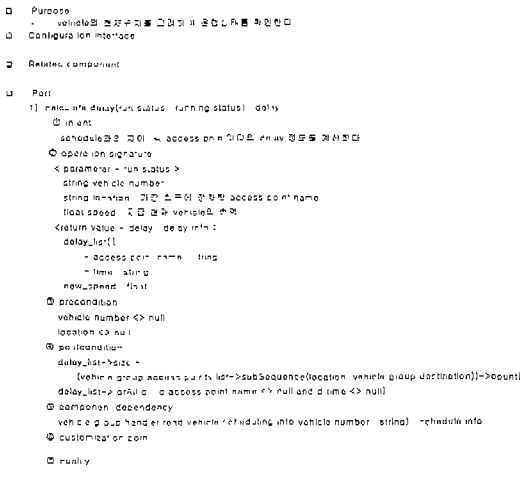

그림 30 delay calculator의 component specification

그림 30은 'delay calculator' 컴포넌트의 component specification을 나타낸다.

'delay calculator' 컴포넌트는 vehicle의 현재위치를 고려하여 운행상태를 확인하는 것을 목적으로 하며, 이를 위해 'calculate delay'라는 인터페이스를 가지고 있다. 그림 31은 'delay calculator'의 행동을 표현하기 위한 state diagram을 나타낸다.

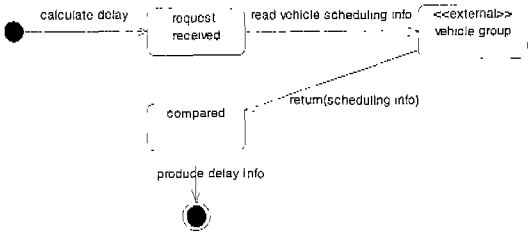

그림 31 delay calculator의 행동

5. 결 론

기존의 컴포넌트 기반 개발 방법론들은 컴포넌트를 정의하고, 표현하는 데 초점을 두었다. 그러나 컴포넌트 기반 개발 방법의 최대 장점인 컴포넌트의 재사용성을 증가시키기 위해서는 관련된 시스템의 집합인 도메인을 충분히 고려하여야 한다. 그런데, 도메인을 체계적으로 분석하고 모델을 생성하려는 기존의 도메인 엔지니어링 방법들은 컴포넌트의 특성을 고려한 도메인 아키텍처 개발 방법에 대해서는 연구가 미흡하였다. 따라서 본 논문에서는 컴포넌트 기반 어플리케이션 개발을 지원하는 도메인 아키텍처에 초점을 두고서, 컴포넌트의 특성을 고려한 도메인 엔지니어링 방법을 제안하였다. 이를 위해 체계적으로 도메인을 정의하고 분석하여 도메인 모델을 개발하는 과정을 정의하였다. 그리고 개발된 도메인 모델을 바탕으로 컴포넌트를 추출하는 방법을 제안하였으며, 컴포넌트와 그들간의 관계를 표현하는 도메인 아키텍처를 개발하는 방법을 정의하였다. 또한 어플리케이션을 개발하는 과정에서 필요한 정보마다 각 뷰들을 두어 간단하고 정확하게 도메인 아키텍처를 표현할 수 있게 하였다. 도메인 아키텍처는 어플리케이션 개발자가 보다 쉽게 이해하도록 하기 위해 UML을 사용하여 시각적으로 표현하였고, 컴포넌트에 대한 정확한 정보 제공을 위하여 텍스트와 UML에서 제공하는 OCL(Object Constraint Language)을 사용하여 기술하였다. 그리고 본 논문에서 제안한 분석 및 설계 방법을 첨단 대중 교통 시스템에 적용하여 제안한 방법이 실제 도메인에 어떻게 적용되는 지 살펴보았다.

향후 연구과제로 도메인 아키텍처에 포함된 컴포넌트 명세서를 바탕으로 컴포넌트의 내부를 설계하고 구현하는 방법이 필요하다. 그리고 설계된 컴포넌트가 도메인 모델에서 제시하는 요구사항을 만족하는 지, 컴포넌트가 올바르게 동작하는 지 검증하는 방법이 필요하다.

참 고 문 헌

[1] SEI in Carnegie Mellon University, "Component-Based Software Development / COTS Integration," http://www.sei.cmu.edu/str/descriptions/cbsd_body.html

[2] Short, K., "Component Based Development and Object Modeling," Sterling Software, 1997.

[3] Ran, A., "Software Isn't Built From Lego Blocks," *Proceedings of the fifth Symposium on Software Reusability*, pp. 164-169, 1999.

[4] D'souza, D. F. and Wills, A. C., *Objects, Components, and Frameworks with UML*, Addison-Wesley, 1998.

[5] Harmon, P., "Visual Modeling Tolls, Case Vendors, and Component Methods," *Component Development Strategies*, June 1999.

[6] Klingler, C.D., "DAGAR: A Process for Domain Architecture Definition and Asset Implementation.." *In Proceedings of ACM TriAda*, pp. 231-245, 1996.

[7] Kang, K. C., "Feature-Oriented Domain Analysis for Software Reuse," *Joint Conference on Software Engineering*, Nov 17-19, pp. 389-395, 1993.

[8] Kang, K. C., Kim, S., Lee J.,and Kim, K., "FORM: A Feature-Oriented Ruse Method with Domain Specific Reference Architectures," Pohang University of Science and Technology(POSTECH), 1998.

[9] Simos, M., "Organization Domain Modeling(ODM) : Formalizing the Core Domain Modeling Life Cycle," SSR'95, pp. 196-205, 1995.

[10] Creps D., Klingler, C., Levine, L., and Allemang, D., "Organization Domain Modeling(ODM) Guide- book Version 2.0," Software Technology for Adaptable, Reliable Systems (STARS), 1996.

[11] SEI in Carnegie Mellon University, "Domain Engineering: A Model-Based Approach," http://www.sei.cmu.edu/activities/domain-engineering/index.html

[12] Jacobson, I., Booch, G., and Rumbaugh, J., *The Unified Software Development Process*, Addison-Wesley, 1999.

[13] Eriksson, H. and Penker, M., *UML Toolkit*, Wiley, 1998.

[14] Kruchten, P., "Architectural Blueprints-The "4+1" View Model of Software Architecture," *IEEE*

Software, pp. 42-50, November 1995.
[15] Han, J., "An Approach to Software Component Specification," *International Workshop on Component-Based Software Engineering*, pp. 97-102 , 1999.
[16] OMG, "Unified Modeling Language Specification Version 1.3," http://www.omg.org, March 1999.
[17] Bennett, S., McRobb, S., and Farmer, R., *Object-Oriented Systems Analysis and Design*, McGRAW-HILL, 1999.
[18] Egyed, A., Mehta, N., and Medvidovic, N., "Software Connectors and Refinement in Family Architectures," USC Center for Software Engineering, http://sunset.usc.edu, 1999.
[19] 고속도로정보통신공단, "ITS란 무엇인가", http://www.hitelecom.co.kr/sayub/yuji/index.htm.
[20] ITS Korea, "ITS 구성", http://www.itskorea.or.kr/main.html.
[21] Casey, R. F., Labell, L. N., LoVecchio, J. A., Ow, R. S., Royal, J. W., Schwenk, J. C., Carpenter, E. J., Moniz, L., Schweiger, C. L., and Marks, B., "Advanced Public Transportation Systems : The State of the Art Update '98," U.S. Department of Transportation, January 1998.

하 현 주

1999년 2월 부산대학교 컴퓨터공학과(학사). 2001년 2월 부산대학교 컴퓨터공학과(석사). 2001년 2월 ~ 현재 삼성전자 정보통신총괄 통신연구소 연구원. 관심분야는 소프트웨어 아키텍쳐, 컴포넌트 기반 소프트웨어 개발 방법론, 도메인 엔지니어링, UML, CASE Tool을 이용한 실시간 시스템 개발, WCDMA 단말 개발 등임.

문 미 경

1990년 2월 이화여자대학교 전자계산학과(학사). 1992년 2월 이화여자대학교 전자계산학과(석사). 2000년 3월 ~ 현재 부산대학교 컴퓨터공학과 박사과정 재학중. 1998년 3월 ~ 1999년 2월 안산1대학 사무자동학과 겸임교수. 관심분야는 도메인 공학, 컴포넌트 기반 소프트웨어 개발, 객체 지향 방법론, 도메인 분석 및 아키텍쳐 개발 등임.

염 근 혁

1985년 서울대학교 계산통계학과(학사). 1992년 Univ. of Florida 전산학과(석사). 1995년 Univ. of Florida 전산학과(박사). 1985년 1월 ~ 1988년 2월 금성반도체 컴퓨터연구실 연구원. 1988년 3월 ~ 1990년 6월 금성사 정보기기연구소 주임연구원. 1995년 9월 ~ 1996년 8월 삼성SDS 정보기술연구소 책임연구원. 1996년 8월 ~ 현재 부산대학교 컴퓨터공학과 조교수. 부산대학교 컴퓨터 및 정보통신 연구소 연구원. 관심분야는 컴포넌트 기반 소프트웨어 개발, 도메인 공학, 소프트웨어 아키텍쳐, 객체지향 소프트웨어 개발방법론, 요구 검증, 분산 컴포넌트, 소프트웨어 재사용 등임.