모델 템플리트를 이용한 도메인 모델 개발과 재사용

Development of Domain Model and Reuse Using Model Template

김 지 홍*
Ji-Hong Kim

요 약

도메인 모델은 객체 모델 개발과 소프트웨어 설계에 관한 결정에 큰 영향을 끼치고 있어, 오늘날 많은 객체지향적 시스템과 컴포넌트 기반의 소프트웨어 개발 시 도메인 모델을 만들고 있다. 그러나 UML과 객체지향적 방법론에서는 재사용을 위한 개발과 재사용을 수반한 개발의 지원이 부족하여 매번 새로이 모델을 개발하고 있으며 프로젝트 개발 기간 지연과 불충분한 모델 생성을 초래하고있다. 이러한 문제는 UML 표기의 확장과 재사용 처리방법을 통해 해결될 수 있다.

본 연구에서는 도메인 모델의 재사용을 위하여 UML 기반의 도메인 모델 템플리트를 설계하고, 분석정보의 재사용을 위한 도메인 모델 개발 방법을 제안하였다. 아울러 제안된 표현을 인터넷 응용에 적용하여 도메인 모델과 도메인 템플리트를 생성할 수 있었다.

Abstract

Since domain model affects largely on the development of object model and design decisions, this model is widely used in the object-oriented and component-based system development. Current OO methods and UML notation, however, do not support both engineering with reuse and engineering for reuse. This problem causes delay in project development time and inadequate domain model. The integration of extended UML notation and reuse process method can provide a solution to the reusability problem.

In this paper, we designed UML based domain model template for the reuse of domain model and proposed domain model development method for the reuse of analysis information. In addition, it was possible to represent reusable domain model template in UML and to develope domain model in the internet sales domain.

1. 서 론

급변하는 기업 및 기술환경에 능동적으로 대응하기 위하여 품질 좋은 소프트웨어를 빠르게 개발하고자 하는 요구는 날로 증가하고 있다. 그러나 소프트웨어 개발은 아직도 시간과 비용이 많이 필요하고 아주 높은 위험성을 갖고 있다. 소프트웨어 프로젝트의 1/3이 취소되고 있으며 단지 1/6만이 성공을 하고 비용과 스케줄은 원래 계획보다 2-3배 더 소요되는 보고도 있다[1].

소프트웨어 분석 단계는 개발하고자하는 시스템의 요구사항을 식별하는 단계로서 소프트웨어의 범위, 성능, 사용 및 유지보수성에 많은 영향을 끼치는 사항들이 결정되는 중요한 단계이다. 이러한 소프트웨어 개발 초기단계에 필요한 것으로 도메인 모델이 있다. 도메인 모델은 개발 요청을 받은 분야의 배경 및 핵심 개념을 객체지향적으로 표현한 것으로서 객체 모델과 소프트웨어 설계에 관한 결정 시 중요한 분석정보이며 객체지향적 분석에서 생성되는 가장 중요한 산출물이다[2]. 이러한 이유로 오늘날 대부분의 객체지향적 시스템 또는 컴포넌트 기반의 소프트웨어 개발 시에는 도메인 모델을 개발하고 있다[3,4].

대부분의 도메인 모델 개발 경우 해당 분야 전문가와 상의, 상세 조사 및 분석이 필요하기 때문에 많은 시간이 걸리며 분석자의 능력에 따라 도메인 모델 개발 기간과 품질에 큰 차이를 보이고

* 경원대학교 컴퓨터공학과 부교수
wiskjh@mail.kyungwon.ac.kr

있다. 그러나 재사용 없이 행해지고 있는 현행 도메인 모델 개발에서는 매번 새로이 작업을 해야 하므로 개발기간이 지연되거나 불충분한 모델 생성의 원인이 되고있다. 따라서 품질 좋은 도메인 모델의 개발은 성공적인 프로젝트 수행의 중요한 요소 가운데 하나라 할 수 있다.

재사용 가능한 도메인 모델을 미리 만들어 두었다가 새로운 응용의 개발 요구가 있을 때 이를 재사용하면 짧은 시간에 경제적인 비용으로 양질의 도메인 모델을 얻을 수 있다. 그러나 오늘날 대부분의 객체지향적 소프트웨어 개발에 가장 많이 사용되는 객체지향 모델링 언어인 UML(Unified Modeling Language)과 객체지향적 분석 및 설계 방법인 RUP(Rational Unified Process)에서는 도메인 모델과 같은 분석정보의 재사용에 관한 표현과 방법의 지원이 부족하다[5]. 이러한 문제는 재사용을 통한 도메인 모델 생성을 지원하는 UML 기반의 재사용 단위와 이를 위한 개발 방법의 제공으로 가능하다.

본 논문에서는 새로운 응용에 필요한 도메인 모델 개발 시 이전의 응용에 개발하였던 도메인 모델을 최대한 재사용하기 위하여 UML을 확장한 모델 템플리트와 점증적 도메인 모델 개발 방법을 제안하고 적용을 보인다. 논문의 제2절에서는 관련 연구로서 분석정보의 재사용, UML 그리고 RUP에 대하여 서술하고, 제3절에서는 분석정보의 재사용을 위하여 도메인 템플리트의 구성 요소와 재사용을 기술하며, 제4절에서는 도메인 템플리트를 기반으로 UML과 RUP를 연계시키는 도메인 모델 개발 방법을 기술하고, 마지막으로 결론을 기술한다.

2. 관련 연구

본 장에서는 분석정보의 한 형태인 도메인 모델의 재사용을 위하여 대표적인 모델링 언어와 객체지향 방법에서의 분석정보 재사용 지원과 문제점을 살펴본다.

2.1 소프트웨어 분석정보의 재사용

분석정보란 소프트웨어 개발 요청을 받고 해결책을 결정하는데 필요한 시스템의 배경, 범위 및 요구사항들을 식별하고 표현한 각종 산출물을 의미한다.

소프트웨어 재사용 접근 가운데 분석정보와 설계정보의 재사용 가능성은 일찍이 Neighbors의 Draco 프로젝트에서 도메인 정보를 분석한 후 도메인 언어로 기술하여 이것을 나중에 재사용하는 방식을 언급하였다. 이는 종래의 코드 재사용에 의한 방법이외에 새로운 가능성을 제공하였다. 지금까지 가장 많이 사용되어온 소프트웨어 재사용 형태는 라이브러리를 통한 원시코드의 재사용이다. 그러나 소프트웨어 개발 시 코딩이 차지하는 비율은 5%밖에 차지하지 않으므로[6] 코딩 단계에서 만의 재사용은 높은 재사용성을 얻을 수 없다. 코드의 재사용 다음으로 고려할 수 있는 접근법은 설계의 재사용이다. 이는 코드의 재사용보다 더 많은 재사용성을 얻을 수 있으나 문제 해결방법이 달라지면 재사용성이 바뀔 수도 있다. 이 보다도 더 높은 재사용성은 분석정보의 재사용이다. 분석정보는 특정 시스템에 요구되는 정보로서 요구사항이 변경되지 않는 한 변화가 없으므로 가장 높은 재사용성을 가질 수 있다. 즉, 분석 단계에서는 코딩 단계에서 발생하는 언어, 운영체제 의존성이 없어진다. 분석정보는 다른 개발 단계에서의 정보보다 가장 고수준의 사용자 요구를 나타낸 것이며, 환경의 변화에 대한 영향이 제일 적은 가장 강력한 재사용 정보 형태이다[7].

그러나 분석정보는 강력한 재사용 정보형태임에도 불구하고 표 1과 같이 소프트웨어 개발 단계

(표 1) 단계별 재사용

개발 단계	분석 단계	설계 단계	코딩 단계
재사용 자산	분석 명세	프레임워크	라이브러리
재사용 노력	고	중	저
재사용 효과	저	중	고

를 비교하여 보면 재사용의 효과는 적은데 비하여 노력은 많이 들고 있어서 분석정보의 부품화 연구가 더욱 필요하다[8].

소프트웨어 부품화를 통한 분석정보의 재사용을 활성화하기 위해서는 부품개발에 따른 부담을 줄이는 노력이 선결 과제이다. 그림 1과 같이 재사용 가능한 소프트웨어 부품 생산에 필요한 비용은 개발 초기에 많이 필요한데 비하여 재사용 효과는 서서히 나타나고 있기 때문에 미리 재사용물을 생성하는데 부담을 갖게된다[9]. 즉, 재사용자의 입장에서는 재사용 가능한 분석정보를 많이 제공받기 원하지만 제공되는 산출물이 충분하지 않아 재사용을 못하고 있으며 재사용 가능한 가공물을 생성하는 입장에서는 재사용의 경제적 부담으로 많은 가공물을 개발하지 못하고 있는 실정이다.

특히, 분석 단계에서 재사용이 가능한 분석패턴은 비즈니스 분야의 객체지향 분석 시 자주 나타나는 거래, 측정, 회계, 조직관계를 다이어그램으로 표현하여 카탈로그로 제공되는 정보이다[10]. 이는 주로 재사용자를 지원하는 분석정보의 한 예라 할 수 있다. 재사용자와 개발자가 함께 분석정보를 손쉽게 이해하고 사용할 수 있도록 표준적 표현과 부품화가 필요하다.

2.2 도메인 모델

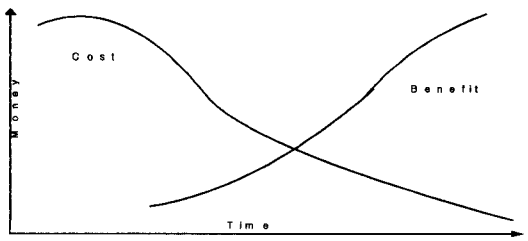
도메인 모델이란 개발하고자 하는 분야의 실세계 객체나 개념적 클래스를 표현한 것으로 개념적 모델, 도메인 객체 모델, 분석 객체 모델이라고도 하며, 객체지향 분석 시 생성되는 가장 중요한 산출물이다[2,11]. 이는 개발에 불필요한 상세 사항은 무시하고 추상화된 관점을 표현하기 때문에 시스템의 핵심적 행위를 담당하는 모든 주요 클래스를 파악할 수 있도록 시각적으로 제공하므로 유사한 시스템 개발 시 해당 도메인을 빠른 시간에 이해하거나 재사용할 수 있는 분석정보이다[12].

도메인 모델은 객체지향적 소프트웨어 개발에서 사용자 인터페이스 설계, 유스케이스 기술, 시스템 내부적 클래스 식별에 사용되는 중요한 정보이다[13]. 특히 적절한 객체나 개념적 클래스의 식별은 문제 도메인 조사의 일부로서, 객체지향적 설계 및 구현에 주요 자료가 되기 때문에 개발하고자 하는 시스템의 성능, 사용성 및 유지보수성과 같은 프로젝트 품질에 막대한 영향을 끼치는 초기 정보이다[14,15].

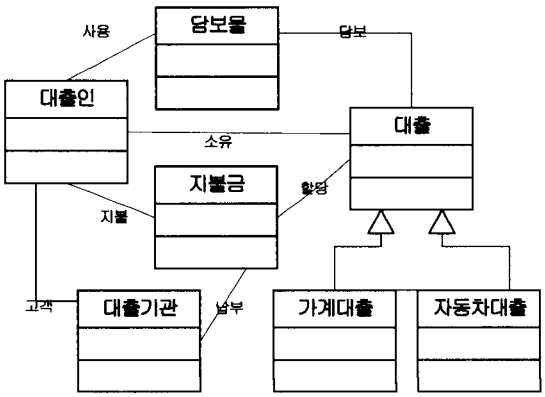
이러한 이유로 도메인 모델은 여러 객체지향적 분석 및 설계와 컴포넌트 기반의 소프트웨어 설계에서 자주 사용되고 있다[2,14]. 그림 2는 도메인 모델의 예로서, 은행 대출업무에 관계되는 개념적 클래스와 그들의 관계로 표현하고있다[14].

도메인 모델의 개발에는 관련분야의 전문가와 의견교환, 상세 분석 및 많은 경험이 필요하다. Larman은 손쉽게 도메인 모델을 작성할 수 있도록 표 2와 같이 4단계 절차를 소개하고 있다[2].

이 방법은 객체 모델링과 비슷한 방식으로, 1단계에서는 후보가 되는 개념을 식별하기 위하여



(그림 1) 재사용 비용과 혜택



(그림 2) 도메인 모델의 예

(표 2) 도메인 모델 작성절차 4단계

1	개념 카타고리 리스트와 명사구 식별 방법을 사용하여 후보 개념의 식별
2	개념을 객체로 표현
3	개념간 연관 관계를 추가
4	개념에 속성을 추가

물리적 객체, 조직체, 장소 등 몇 가지 카타고리 리스트를 이용하여 후보 개념을 식별하거나 문제 도메인 설명서에 나타난 문장의 품사를 이용하여 결정한다. 2단계에서는 식별된 개념을 하나의 객체로서 표현하고 3단계에서는 객체로 표현된 개념들간의 의미 부여를 위하여 관계를 표현한다. 4단계에서는 식별된 객체에 필요한 속성을 추가한다.

도메인 객체를 식별하는데 처음부터 카타고리 리스트나 명사구를 이용하는 식별 방법이외에 이전의 개발 시 작성된 유사 도메인 객체를 재사용할 수 있다면 생산성을 더 높일 수 있다.

2.3 UML과 RUP

UML은 오늘날 객체지향적 분석 및 설계의 실질적 표준 모델링 언어로 인식되고 있으며 RUP는 UML을 이용한 소프트웨어 개발에 가장 많이 사용되고 있는 객체지향적 개발 방법론이다. 본 절에서는 도메인 모델의 재사용 관점에서 UML의 클래스 표현과 RUP의 절차를 살펴본다.

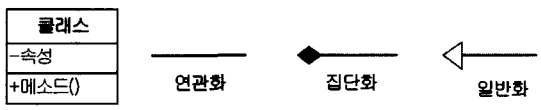
(1) UML과 템플리트 클래스

UML은 미국의 래쇼날 소프트웨어 사에서 객체지향 분석 및 설계 방법 가운데 많이 알려진 Booch의 OOD(Object Oriented Design) 방법, Jacobson의 OOSE(Object Oriented Software Engineering), Rumbaugh의 OMT(Object Modeling Technique) 방법의 장점을 통합하여 개선한 객체지향 모델링 언어로서 OMG(Object Management Group)에서 표준 모델링 언어로 채택한 이후 객체지향 모델링의 실질적 표준으로 인식되고 있다[16]. UML은 객체지향적 분석과 설계에 있어서 시스템의 정적인 표현을 위하여 클래스 다이어그램, 객체 다이어그램, 컴포넌트 다이어그램, 배치 다이어그램이 사용되고 있으며 시스템의 동적인 표현을 위하여는 유스케이스 다이어그램, 순차 다이어그램, 협동 다이어그램, 상태 다이어그램, 활동 다이어그램이 사용된다. UML을 이용한 도메인 모델링에는 그림 3과 같은 클래스 다이어그램과 클래스들간의 관계가 사용되며 메소드 부분은 주로 표현에 사용하지 않는다.

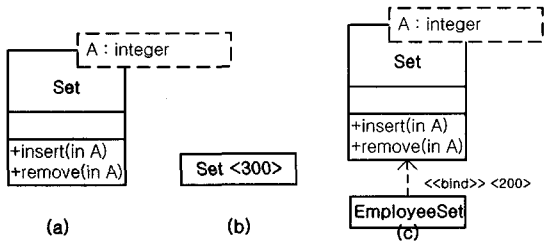
클래스는 응용 도메인에서 동일한 속성과 연산을 공유하는 객체들의 집합으로 이름, 속성, 메소드로 표현된다. 클래스는 시스템에 무엇이 존재하는지를 표현하며 관계는 이들이 어떻게 구조화되어 있는지를 나타낸다. 속성은 클래스에 있는 모든 객체들이 공유하는 것으로 특정 개념을 위한 다양한 값을 표현할 수 있어 도메인 모델의 배경을 이해하는데 도움이 된다. 메소드는 다른 객체가 요청할 수 있는 서비스를 구현한다.

UML에서는 하나의 클래스 정의를 위한 다이어그램 이외에 속성과 메소드를 공유하는 클래스 패밀리를 정의할 수 있도록 다른 형태의 다이어그램을 지원한다. 템플리트 클래스는 유사한 클래스를 위하여 매개변수를 지원하는 클래스이다[17].

템플리트 클래스는 그림 4-a와 같이 클래스의



(그림 3) 클래스와 관계

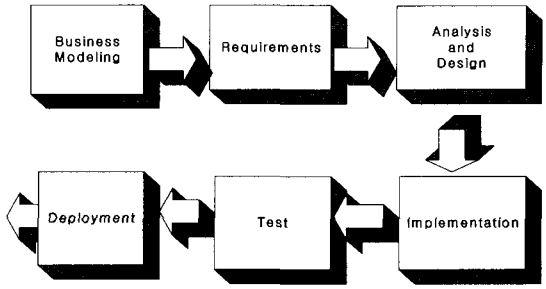


(그림 4) UML의 템플리트 클래스

우측 상단에 파라미터를 표현할 수 있는 부분이 추가되어 실 매개변수의 표기를 지원하며 파라미터 클래스라고 부른다. 파라미터 부분에 사용할 수 있는 매개변수의 형은 클래스, 정수, 논리형과 같은 원시형(primitive type)이 가능하다. 템플리트는 직접 사용할 수 없고, 사용하려면 인스탄스화(instantiation) 하여야 한다. 템플리트를 인스탄스화 하는 방법은 두 가지가 있다. 하나는 그림 4-b와 같이 이름을 갖는 클래스를 선언함으로서 묵시적으로 선언하는 방법과 다른 방법은 그림 4-c와 같이<<bind>> 스테레오타입을 갖는 종속관계를 사용함으로써 명시적으로 표현할 수 있다. UML에서 템플리트 클래스는 이름과 표기의 일부를 C++에서 빌려왔으며 구현에 필요한 템플리트를 모델로 표현하거나 C++로 구현된 코드를 모델로 표현하는 역공학을 지원하는데 주로 사용된다[18]. 템플리트는 다양한 형의 표현을 통하여 구현에서의 융통성은 지원되지만 분석 단계에서 필요한 시스템 배경과 개념 표현을 위한 재사용 지원은 부족하다.

(2) RUP

RUP는 UML을 개발한 래쇼날 소프트웨어 사의 Booch, Jacobson, Rumbaugh에 의해 제안된 객체지향적 소프트웨어 개발 생명주기 모델로서 줄여서 UP(Unified Process)라고도 부른다[19]. 통합 소프트웨어 개발 방법은 주로 단일 시스템의 개발을 지원하고 있으며 재사용을 위한 개발과 재사용을 통한 개발에 대한 지원이 부족하다[5]. 이 방법은 나선형 소프트웨어 개발 모델과 비슷하게 하나의 프로세스는 여러 주기로 구성되며 각 주기의 끝에는 산출물이 제공된다. 각 주기는 도입(inception), 정련(elaboration), 구축(construction), 전이(transition)와 같이 4개의 단계로 이루어지며 각 단계는 반복적인 특성을 갖고 있다. 산출물은 각 활동의 결과로서 나타나는데 기술적 산출물로는 요구사항 집합, 설계 집합, 구현 집합, 배치 집합을 갖는다. 이 가운데 분석 활동이 두드러진 요구



(그림 5) 핵심적(core) 프로세스

사항 집합은 시스템이 해야할 것을 설명한 것으로 유스케이스 모델, 도메인 모델, 분석 모델이 여기에 포함된다.

특정 모델을 얻기 위해 필요한 일련의 활동들을 워크플로우라고 부르는데 RUP에는 그림 5와 같이 6개의 핵심적 워크플로우가 있다. 비즈니스 모델링 프로세스 워크플로우에서는 조직체에 대한 전체적인 이해를 위하여 구조와 동적인 점을 설명하고, 요구사항 프로세스에서는 요구사항 식별을 위하여 유스케이스 기반의 방법을 기술한다. 분석과 설계 프로세스에서는 여러 아키택쳐 관점을 표현하며 구현, 시험, 배치 프로세스에서는 소프트웨어의 개발, 시험, 현장 배치를 다룬다.

RUP에서 도메인 모델은 첫 번째 핵심적 프로세스인 비즈니스 모델링에서 생성되는 산출물 중 하나로서 도메인 객체와 개념적 클래스, 개념적 클래스간의 연관화 관계, 개념적 클래스의 속성으로 나타낸다. 래쇼날 통합 프로세스 방법에서는 도메인 모델의 필요성과 사용은 강조하고 있으나 구체적인 개발 절차는 언급하지 않고 있다. 대신에 인터뷰나 요구명세서 검토 그리고 분석가와 도메인 전문가를 팀으로 하는 워크숍을 통한 획득을 제시하고 있다. 특히 RUP에서 도메인 모델은 비즈니스 모델의 단순화된 변형으로 간주하고 비즈니스 모델의 분석으로 도메인 모델의 추출을 언급하고 있다[6].

관련연구를 통하여 볼 때 분석정보는 가장 강력한 재사용 대상이 될 수 있으나 아직까지 소프트웨어의 품질과 생산성 향상을 위한 재사용은 주로

코딩과 설계 단계에 국한되어있다. 소프트웨어 분석정보의 한 형태인 도메인 모델은 개발하고자 하는 분야의 배경 및 개념을 제공하기 때문에 많은 객체지향시스템 이나 컴포넌트 기반의 소프트웨어 개발의 초기단계에 이 모델을 만들고 있다.

그러나 기존의 객체지향적 개발 방법이나 UML은 분석정보의 재사용에 대한 지원이 부족하여 새로운 응용을 위한 도메인 모델이 필요할 때마다 이전의 산출물을 재사용하지 못하고 분석가는 하나의 시스템만을 위한 도메인 모델을 새로이 만들고 있어서 개발시간의 지연, 불충분한 모델 생성, 비용 증가의 문제를 야기하고 있다. 이러한 문제 해결에는 도메인 모델의 재사용과 생성에 공통적 표현을 위하여 UML을 확장한 도메인 모델 템플리트와 재사용 공급자와 재사용자간의 유기적 활동을 위한 진화적 도메인 모델 개발 방법이 필요하다.

3. 도메인 모델 개발을 위한 모델 템플리트 구성 요소와 재사용

본 장에서는 도메인 모델의 생성과 재사용 작업에 함께 표현하고 처리할 수 있도록 UML을 확장한 도메인 모델 템플리트를 제안한다.

3.1 도메인 모델 템플리트와 템플리트 종류

(1) UML과 모델 템플리트

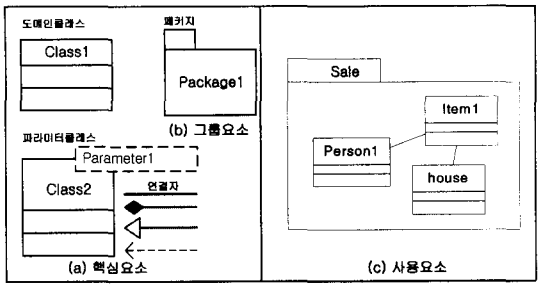
도메인 모델 템플리트란 특정 도메인의 업무처리에 자주 사용되는 개념 및 내용을 쉽게 단위적으로 표현하고 재사용을 함께 지원하는 틀로서 모델 템플리트 또는 도메인 템플리트라고도 부른다. 틀은 새로운 템플리트를 정의하거나 틀을 재사용하여 도메인 모델을 만드는데 사용한다. 모델 템플리트는 분석정보를 쉽게 이해, 표현 및 사용하여 재사용을 위한 표현과 처리를 효율적으로 지원할 수 있도록 핵심, 그룹, 사용의 3가지 요소를 제공한다. 핵심요소는 객체지향적 모델링 표현에, 그룹요소는 특정 부분의 표현을 하나로 묶기를, 사용요소는 도메인 모델 또는 템플리트의 정의를 지원한다.

체계적으로 주요 3개 요소의 사용을 지원하는 도메인 모델 템플리트는 표 3과 같이 정의된다. 핵심요소(<Core_Element>)는 클래스(<Class>)와 연결자(<Connector>)를 가지며 그룹요소(<Group_Element>)는 연결자로 이어지는 2개 이상의 클래스 모음을 나타낸다. 템플리트(<Template>)는 단일의 그룹요소 또는 연결자로 이어지는 2개 이상의 그룹요소 모음으로 표현된다.

도메인 템플리트 구성 규칙에 있는 주요 요소들은 그림 6과 같이 UML을 이용하여 시각적으로 표현된다. 핵심요소에 있는 클래스들은 템플리트 클래스와 클래스 다이어그램으로, 연결자는 연관화, 집단화, 일반화, 의존 관계로 표현된다. 그룹요소는 패키지 다이어그램으로 나타내고 사용요소는 핵심요소와 그룹요소를 지원하는 UML 다이어그램으로 표현된다.

(표 3) 도메인 템플리트 구성 규칙

<Core_Element>	=	<Class> | <Connector>
<Class>	=	<Domain_Class> |
		<Parameter_Class>
<Connector>	=	<Association> | <Aggregation> |
		<Generalization> | <Dependency>
<Group_Element>	=	<Class> <Connector> <Class> |
		<Group_Element><Connector><Class>
<Template>	=	<Group_Element> | <Template>
		<Connector> <Group_Element>
<Domain_Class>	=	/* UML로 표현되는 도메인 클래스 */
<Parameter_Class>	=	/* UML로 표현되는 파라미터클래스*/
<Association>	=	/* UML로 표현되는 연관화 관계 */
<Aggregation>	=	/* UML로 표현되는 집단화 관계 */
<Generalization>	=	/* UML로 표현되는 일반화 관계 */
<Dependency>	=	/* UML로 표현되는 의존 관계 */



(그림 6) UML로 표현한 도메인 모델 템플리트

사용자 입장에서 도메인 모델 템플리트에 있는 핵심요소와 그룹요소는 모델링을 위해 제공되는 몇 개의 UML 심볼로 간주되며 사용요소는 사용자 정의의 도메인 모델이나 템플리트를 만드는 작업 영역으로 인식된다. 새로운 템플리트나 도메인 모델을 만들 때 사용요소 영역은 재사용 여건에 따라 0 또는 1이상의 재사용물을 제공받아 핵심 및 그룹요소로 추가나 합성 작업을 지원한다.

(2) 분석정보 재사용을 위한 수직적 모델 템플리트와 수평적 모델 템플리트

템플리트에 있는 UML로 표현되는 핵심요소와 그룹요소 확장에 필요한 사항을 식별하기 위하여 분석정보 재사용이 가능한 템플리트의 종류를 살펴본다. 시스템 배경 및 개념에 관한 정보는 일반성과 특성에 따라 특정 도메인에서만 재사용이 불가피한 부분과 여러 다른 도메인에서 공통적으로 재사용 할 수 있는 부분이 있다. 도메인 모델 템플리트는 두 가지 경우를 모두 지원할 수 있도록 표 4와 같이 수직적 도메인 템플리트와 수평적 도메인 템플리트를 지원한다.

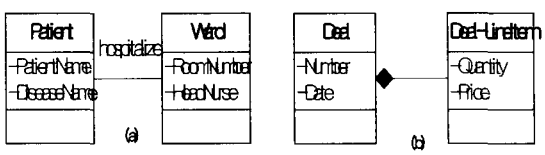
수직적 도메인 모델 템플리트는 철도 도메인, 부동산 도메인, 특정 게임 도메인과 같이 다른 업무와 중복되지 않고 고유한 업무를 갖는 특정 분야에서의 재사용을 지원하는 템플리트이다. 수직적 도메인 템플리트에서는 재사용하고자 하는 분야와 재사용을 위해 제공받을 수 있는 분야가 동일한 경우에는 가장 큰 재사용성을 얻을 수 있는 형태이다. 그러나 템플리트의 내용이 특정 도메인에 국한되어 있기 때문에 분야가 다르면 재사용 할 수 없게되어 재사용 대상의 폭이 좁은 단점을 갖고 있다. 그림 7-a는 수직적 도메인 모델 템플리트의 한 예로서 환자이름(PatientName)과 병명(DiseaseName) 속성을 갖고 있는 환자(Patient)가 입원실번호(RoomNumber)와 담당간호사(Head-Nurse)속성을 갖는 입원실(Ward)에 입원(hospitalize)하고있는 의료분야(healthcare) 도메인의 배경을 나타내고 있다. 수직적 도메인 모델 템플리트의 재사용성을 높이기 위해서는 하나의 큰 템플리트보다는 연결이 가능한 여러 개의 모듈화된 템플리트의 지원이 필요하다.

수평적 도메인 모델 템플리트는 신용조회, 재고관리와 같이 분야가 다른 여러 도메인에서 공통적으로 자주 사용되는 부분을 재사용 할 수 있도록 준비한 템플리트이다. 수직적 템플리트보다는 여러 응용에 재사용 될 수 있는 장점을 갖지만 한번에 재사용되는 크기는 상대적으로 수직적 템플리트 보다 작다.

수평적 도메인 모델 템플리트의 재사용성을 높이기 위해서는 개발하고자 하는 도메인에 적용이 쉬운 일반화된 다양한 종류의 템플리트 제공이 중요하다. 그림 7-b는 수평적 도메인 모델 템플리트의 한 예로서 주문 접수 서브 도메인의 일부를 표현한 것이다. 홍정번호(Number)와 홍정일자(Date) 속성을 갖는 홍정(Deal)은 개수(quantity)와 가격(Price) 속성을 갖는 홍정세부항목(Deal-LineItem)들로 구성

(표 4) 도메인 템플리트의 종류

1	수직적 도메인 모델 템플리트	특정 도메인에서만 재사용이 가능한 배경과 개념을 표현
2	수평적 도메인 모델 템플리트	다른 도메인에서 재사용이 용이한 범용적 배경과 개념을 표현



(그림 7) 수직적 템플리트와 수평적 템플리트 표현

된 집단화 관계를 나타내고 있다. 홍정(Deal)-홍정 세부항목(Deal-LineItem)과 같은 수평적 템플리트는 특정 도메인에 관계없이 주문, 판매 등과 같은 다양한 형태의 홍정과 이에 따른 세부항목의 표현이 필요한 도메인에는 모두 적용이 가능하다.

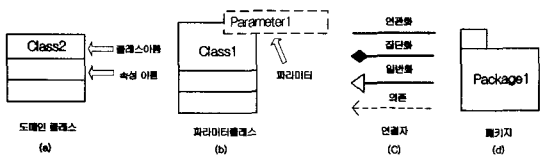
분석정보 재사용을 지원하는 수직적 모델 템플리트와 수평적 모델 템플리트의 표현과 재사용 처리는 UML의 확장으로 가능하다.

3.2 재사용을 지원하는 도메인 모델 템플리트 구성 요소

도메인 템플리트는 재사용자의 관점과 템플리트 개발자의 관점을 모두 충족되어야 한다. 개발자는 손쉽게 도메인 템플리트를 만들 수 있어야 하고 사용자는 손쉽게 재사용할 수 있어야 하므로 이해와 표현이 용이한 UML 요소의 확장이 유리하다. 도메인 템플리트는 분석가가 빠른 시간에 도메인 모델을 얻을 수 있도록 그림 8과 같이 도메인 클래스, 파라미터 클래스, 연결자, 패키지로 표현된다.

도메인 클래스는 특정 도메인에서의 핵심적 개념과 용어를 구체적으로 표현하는데 사용되며 그림 8-a와 같이 클래스 이름, 속성, 메소드 3부분으로 표현된다. 도메인 클래스는 도메인 모델 표현에 가장 많이 사용되는 템플리트 구성 요소이다. 도메인 모델은 특정 분야의 개념을 나타내므로 도메인 클래스의 메소드 부분은 사용하지 않고 클래스 이름과 속성만으로 표현한다.

파라미터 클래스는 그림 8-b와 같이 기존의 도메인 클래스에 파라미터 부분이 추가된 형태를 갖는 구조로서 새로운 환경에 적절한 매개변수를 적용할 수 있는 구조이다. UML의 파라미터 클래스는 구현단계에서 C++와 같은 언어로 코딩할 때 응용에 맞는 유형의 손쉬운 선택을 지원하기 위하여 클래스 다이어그램에 형을 매개변수화 시켰다. 그러나 도메인 모델은 시스템 배경의 표현이 목적이므로 메소드와 구현의 상세한 표현은 필요하지 않다. 오히려 도메인 개념을 모델링 할 수 있도록 풍부한 속성의 표현이 필요한데 이러한 문제는 파라미터 클래스의 확장으로 가능하다. 템플리트 클래스의 매개변수는, 예를 들어, "Item:Book"과 같이 "형식 매개변수:실 매개변수"의 형태로 표현된다. 표 5는 분석정보의 재사용을 위한 클래스의 파라미터 규칙을 나타낸다.



(그림 8) 템플리트 구성 요소

<FormalName>은 형식 매개변수로서 이미 UML로 정의된 클래스나 속성 이름을 나타내고 <UserDefined Name>은 실 매개변수로서 새로운 모델에 필요한 이름을 기술한다. 형식 매개변수에 클래스 이름을 사용할 필요가 있는 경우에 가독성을 위하여 클래스 이름 표현 대신에 키워드 "NAME"을 사용할 수 있다. 하나 이상의 매개변수가 필요할 때는 ","로 구분된다. 예를 들어 매개변수 "Item:Book, Number:ISBN"은 Number라는 속성을 갖는 클래스 Item을 재사용하여 ISBN이라는 속성을 갖는 클래스 Book의 생성을 나타낸다. 이는 "NAME:Book, Number:ISBN"과 동등한 의미를 갖는다.

(표 5) 템플리트 클래스의 파라미터 규칙

<TemplateParam>	=	<Parameter> |
		<TemplateParam> ","
		<Parameter>
<Parameter>	=	<FormalName> ":"
		<UserDefinedName>
<FormalName>	=	<Keyword> | <ClassName> |
		<AttributeName>
<UserDefinedName>	=	/*사용자 정의의 실 매개 이름*/
<ClassName>	=	/* 클래스 이름 */
<Keyword>	=	"NAME"
<AttributeName>	=	/* 속성 이름 */

파라미터 클래스를 UML에 의하여 시각적으로 자세히 나타내면 그림 9-a처럼 기존의 클래스와 속성 이름은 파라미터 클래스에서 형식 클래스 이름과 형식 속성 이름이라 부르고 파라미터는 그림 9-b처럼 형식 매개변수와 실 매개변수로 표현된다.

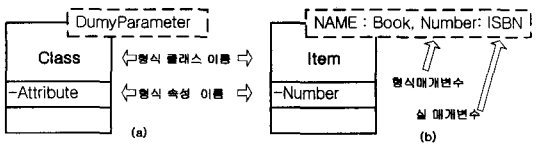
연결자는 특정한 의미의 분석정보를 나타내도록 해당분야의 도메인 클래스나 템플리트 클래스들 사이에 관계 표현에 사용된다. 이는 템플리트 정의에 사용될 뿐만 아니라 도메인 모델 생성 시 재사용을 위한 관계의 표현에도 사용된다. 그림 8-c는 도메인 템플리트에서 사용될 수 있는 연결자를 보인다

패키지는 그림 8-d와 같이 도메인 모델의 일부 내용을 하나의 단위로 그룹화 시켜 표현하는 메카니즘이다. 도메인 배경이 커지게 되면 이에 따르는 분석정보의 이해가 차츰 힘들어지고 재사용이 어려워지기 때문에 여러개의 작은 서브도메인으로의 표현이 유리하다.

지금까지 설계한 도메인 모델 템플리트는 UML로 표현할 수 있어 범용적 표현과 이해성을 제공하며 도메인 클래스, 파라미터 클래스 그리고 연결자와 같은 기본 구조는 도메인 모델 개발을 위한 생성 활동과 재사용 활동에 공통적인 틀을 지원할 수 있는 장점을 함께 갖는다.

4. 템플리트 기반의 진화적 도메인 모델 개발 방법과 재사용

본 장에서는 3장에서 설계한 UML 기반의 도메인 모델 템플리트를 이용하여 도메인 모델을 개발하는 절차와 재사용 방법을 살펴본다.

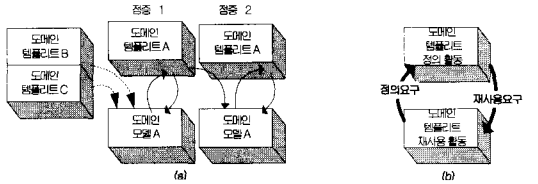


(그림 9) 템플리트 클래스와 매개변수

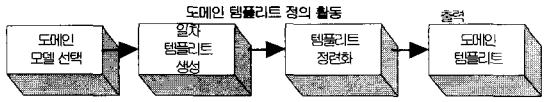
4.1 진화적 도메인 모델 개발 방법

진화적 도메인 모델 개발 방법이란 빠른 시간에 품질 좋은 도메인 모델을 경제적인 비용으로 개발하기 위하여 UML을 기반으로 도메인 모델과 도메인 모델 템플리트를 반복 점증적으로 개발을 지원하는 프로세스이다. 이 방법은 그림 10-a와 같이 이전에 만든 재사용물 가운데 새로운 요구에 사용할 수 있는 산출물을 재사용하여 도메인 모델을 개발하고 이를 다시 동등한 의미를 갖는 도메인 템플리트로 만드는 일련의 반복(iteration) 작업으로 새로운 요구가 매번 추가되며 점증(increment)적으로 개발된다. 따라서 시스템 개발이 많아질수록 새로운 시스템 개발 요구가 반영된 도메인 모델과 템플리트는 점증 1, 2, ...n의 형태로 만들어지며 차츰 범용적인 재사용 가능한 분석정보 모델로 진화(evolution)한다. 모델 개발 방법은 그림 10-b처럼 크게 도메인 템플리트 정의활동과 도메인 템플리트 재사용활동의 반복적 작업으로 이루어진다.

도메인 템플리트 정의 활동이란 도메인 모델 개발을 위한 재사용물을 개발하는 것으로 진화적 개발의 점증에 개발된 산출물들을 재사용하여 3.1절에서 식별한 수직적 도메인 모델 템플리트 또는 수평적 도메인 모델 템플리트를 개발하는 활동이다. 도메인 템플리트 정의 활동은 그림 11과 같이 도메인 모델 선택, 일차 템플리트 생성과 템플리트 정련화(refinement)와 같이 연관된 작업들로 이루어진다.



(그림 10) 진화적 도메인 모델 개발 방법



(그림 11) 도메인 템플리트 정의 활동

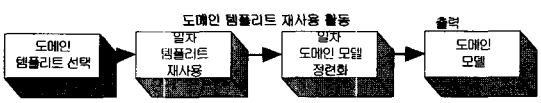
도메인 모델 선택은 현 점증의 도메인 모델을 만든 후 일반화된 템플리트를 생성하는데 재사용할 대상을 선정하는 활동이다. 요구사항을 검토하고 현재의 점증에서 개발한 도메인 모델과 이전 유사 시스템의 점증에서 재사용 가능한 산출물을 식별한다. 일차 템플리트 생성에서는 도메인 템플리트의 일차적 구조를 결정하기 위하여 선택한 후보 도메인 모델을 UML 기반의 재사용 가능한 템플리트로 정의하며 4.2절 (2)항에서 언급한다. 템플리트 정련화는 일차적 도메인 템플리트의 생성 이후 불필요한 템플리트 요소를 삭제하거나 필요 요소를 추가하기 위하여 처리중인 클래스 요소와 관계의 이름을 배경에 맞도록 하는 작업들이 수행된다.

도메인 템플리트 재사용 활동은 새로운 프로젝트에 필요한 도메인 모델을 얻기 위하여 이전에 만든 도메인 템플리트를 재사용하는 작업들로 이루어진다. 체계적인 재사용 활동을 위해서는 그림 12와 같이 도메인 템플리트 선택, 일차 템플리트 재사용 활동, 일차 도메인 모델 정련화 작업이 필요하다.

도메인 템플리트 선택에서는 프로젝트 목적 설명서를 검토하고 필요한 도메인 템플리트를 결정한다. 일차 템플리트 재사용은 이전 프로젝트에서 개발한 도메인 모델 템플리트에 4.2절 (1)항에 언급되는 템플리트 재사용 연산을 적용하여 윤곽적인 도메인 모델을 생성하는 작업이다. 일차 도메인 모델 정련화는 일차 재사용 활동으로 나타나는 불필요한 템플리트 요소의 삭제나 필요 요소를 추가하고 표현된 클래스나 속성 또는 관계의 이름을 요구사항에 맞도록 세련을 수행한다.

4.2 도메인 템플리트의 재사용과 생성

본 절에서는 진화적 도메인 모델 개발의 주요 프로세스인 템플리트 재사용과 생성을 위해서 UML을 이용한 분석정보의 재사용과 개발의 각 점증에 나타난 산출물을 이용한 템플리트 생성을 살펴본다.



(그림 12) 도메인 템플리트 재사용 활동

(1) 일차 템플리트 재사용

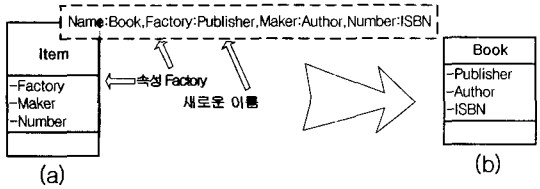
일차 템플리트 재사용이란 새로운 요구사항에 맞는 도메인 모델을 얻기 위한 도메인 템플리트 재사용의 2번째 작업으로 클래스나 패키지에 재사용 연산을 수행하여 일차적 모델을 획득하는 활동이다. 재사용을 위해서 먼저 선택된 템플리트와 요구사항을 검토하고 재사용 하고자하는 분석정보의 크기에 따라 도메인 클래스나 파라미터 클래스 또는 패키지를 선택하고 표 6과 같은 재사용 연산을 수행하여 도메인 모델을 생성한다.

파라미터 적용에서는 분석정보를 위한 템플리트 클래스를 재사용하여 새로운 도메인 모델에 필요한 도메인 객체를 생성한다. 먼저, 선택한 후보 클래스의 속성과 요구사항을 분석하고 기존 도메인 클래스에서 약간의 속성의 변경만으로 새로 요구를 만족되는 경우, 응용에 필요한 값을 매개변수로 적용한다. 그림 13-a는 대여 도메인 표현에 사용되는 파라미터 클래스 Item에 적용되는 실 매개변수를 보이고 있다.

매개변수에는 클래스 이름의 적용을 위해 형식 매개변수로 키워드 파라미터인 NAME과 실 매개변수 Book이 사용되었으며 속성을 위한 형식 매개

(표 6) 도메인 모델 템플리트 재사용 연산

1	파라미터적용	정의된 파라미터 클래스 재사용을 위해 매개변수를 적용
2	인스탄스화	새로운 도메인 객체나 패키지를 생성
3	합성화	두 개 이상의 템플리트 구성 요소의 합성화



(그림 13) 재사용을 위한 파라미터 적용

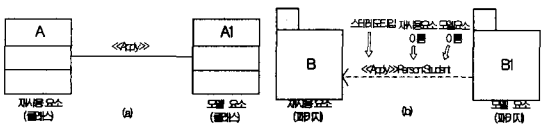
변수 Factory에 대응되는 실 매개변수로 Publisher가 적용되었다. 그림 13-b는 파라미터 적용 연산의 결과로서 출판사(Publisher), 저자(Author), 등록번호(ISBN)와 같은 속성을 갖는 책(Book) 도메인 객체의 생성을 나타내고 있다.

인스탄스 연산은 도메인 클래스나 패키지를 재사용하여 새로운 응용에 필요한 도메인 객체나 패키지를 생성한다. 이는 몇 개의 속성 변경이 아니라 클래스나 패키지 전체의 재사용을 지원한다. 표 7과 같이 인스탄스 연산의 표현 <Apply_Instance>는 재사용 할 요소와 새로 생성할 요소사이에 "≪Apply≫" 또는 ""≪Apply≫" <Items>"관계로 표현된다. 그림 14-a는 도메인 클래스 A를 재사용하여 도메인 객체 A1의 생성을 나타내고 있다.

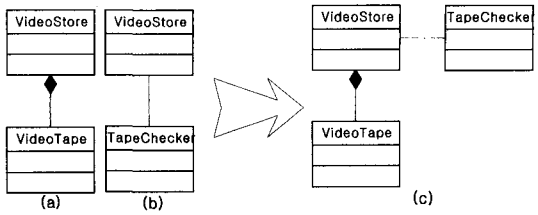
패키지는 도메인 클래스보다 좀더 큰 크기의 분석명세의 재사용을 지원한다. 패키지는 여러 모델 요소를 갖고있으므로 패키지를 재사용하는 경우, 내부에 있는 모델 요소를 응용에 적절한 이름으로 변경하면서 새로운 패키지 생성을 표현할 수 있다. 이 경우에 항목 <Items>은 ""≪Apply≫" <Reuse_Element> ":" <New_ Element>"형식이 사용되는데 재사용 요소를 나타내는 <Reuse_Element>와 새로운 요소를 나타내는<New_Element>는 한 쌍을 이루어 표현된다. 패키지 내부 항목에서 두

(표 7) 재사용을 위한 인스탄스화 규칙

<Apply_ Instance>	=	"<<Apply>>" | "<<Apply>>" <Items>
<Items>	=	<Reuse_Element> ":" <New_Element>
<Ins_List>	=	<Items> | <Items> "," <Ins_List>
<Reuse_ Element>	=	/*템플리트에 있는 클래스 또는 패키지*/
<New_ Element>	=	/*새 모델에 생성될 클래스 또는 패키지*/



(그림 14) 재사용을 위한 인스탄스



(그림 15) 재사용을 위한 합성화

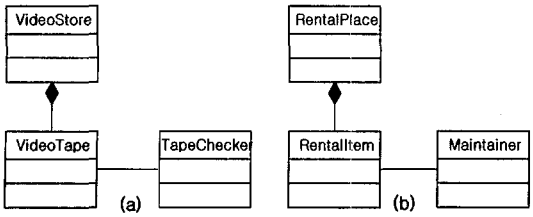
개이상의 변경이 필요한 경우 이들 항목사이를 "," 로 구분하여 반복적으로 표현한다. 그림 14-b는 패키지 B에 있는 모델 요소 Person을 Student로 변경하면서 새로운 패키지 B1의 생성을 표현하고 있다.

합성화 연산은 2개 이상의 도메인 템플리트 재사용 시 서로 다른 템플리트 모두에 존재하는 클래스나 패키지를 중첩하고 연관된 구성 요소를 서로 연결함으로서 새로운 도메인 모델의 생성과 재정의를 지원한다. 구성 요소간의 합성은 연관화, 집단화 및 일반화 관계로 연결될 수 있다. 그림 15는 비디오 대여점에서 취급하는 임대물과 대여점에 근무하는 직원을 나타내는 템플리트 구성 요소 a와 b를 합성한 결과를 나타낸다.

(2) 일차 템플리트 생성

일차 템플리트 생성은 도메인 템플리트 정의의 2번째 활동으로 지금까지 개발된 특정 도메인의 산출물을 이용하여 차후 유사 시스템 개발 때 더 많이 재사용할 수 있도록 재사용물, 즉 도메인 모델 템플리트를 재 정의하는 활동이다. 생성 작업에서는 기존 도메인 모델의 일반화, 파라미터 정의, 그룹화 작업을 통하여 재사용 가능한 도메인 템플리트를 정의한다.

분석정보의 재사용을 위한 도메인 클래스의 일반화란 도메인 개념을 나타내는 클래스와 속성들이 여러 도메인 모델 표현에 용이하도록 공통점을 식별하고 표현하는 행위를 말한다. 진화적 도메인 모델 개발은 점증적 개발을 지원하므로 몇 번의 점증에 나타난 클래스들의 공통점 식별은 일반화된 클래스 이름과 속성의 정의를 지원한다.

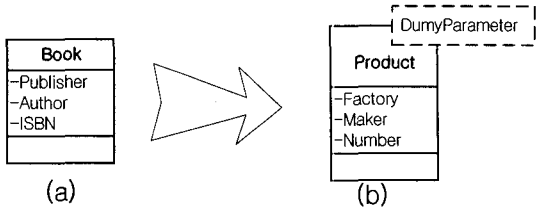


(그림 16) 템플리트 사용요소의 일반화 정의

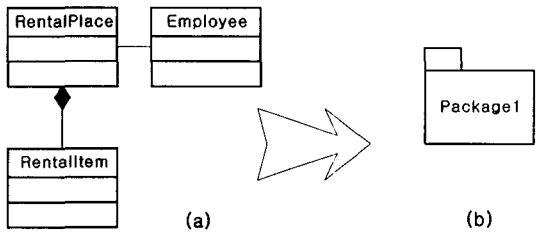
그림 16-a는 특정 점증에서 얻은 비디오 대여점에 관한 도메인 모델의 일부이다. 그림 16-b는 비디오 대여 도메인을 포함한 몇 개의 점증들의 공통점을 식별하고 일반화 시켜 여러 종류의 임대 도메인에 재사용 할 수 있도록 표현된 클래스 임대장소, 임대항목 그리고 유지 보수자를 보인다.

파라미터 정의는 유사한 시스템의 진화적 개발 점증들에 나타난 여러 도메인 클래스 이름과 속성들을 분석하여 특성들의 변이(variation)를 형식 클래스와 형식 속성 이름으로 표현하는 행위를 말한다. 식별된 분석정보의 공통점은 해당 도메인 템플리트의 기본적 개념으로 사용되며 변이점은 특정 요구에 적용 가능한 개념을 나타낸다. 그림 17은 이전의 도서관 프로젝트에서 개발한 도메인 클래스 Book을 여러 응용에 취급할 수 있는 형식 클래스 Product로 재 정의를 보인다. 그림 a에 있는 도메인 클래스의 속성 Publisher, Author, ISBN이 그림 b에서는 형식 속성 Factory, Maker, Number를 갖는 파라미터 클래스로 정의되었다.

그룹화는 재사용을 위한 표현과 처리가 용이하도록 도메인 모델을 특정의 기준에 따라 논리적 단위로 정의하는 행위이다. 각 점증에 나타난 도메인 모델들을 분석하여 공통점, 변이점을 식별하거나 개념적 또는 논리적 관점으로 단위화하고 분류하여 UML의 패키지 다이어그램으로 표현한다. 특히 그룹화는 대규모 시스템의 도메인 모델 개발 시 발생하는 표현과 크기의 복잡도를 관리할 수 있도록 여러개의 논리적인 도메인으로 분리를 지원하여 재사용을 위한 이해와 처리를 용이하게 해준다. 그림 18은 물품 판매에 관련된 점포, 물건, 직원으로 표현한 대여 도메인의 부분적



(그림 17) 템플리트 사용요소의 파라미터 정의



(그림 18) 템플리트 사용요소의 그룹화

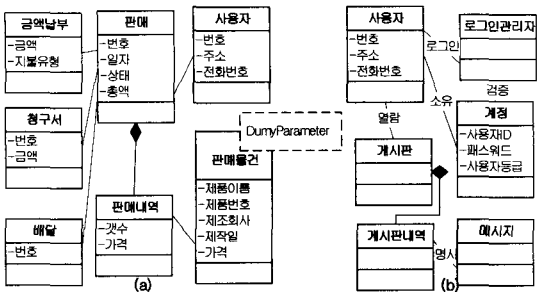
개념을 하나의 패키지화시킨 예이다.

UML 기반의 도메인 모델 템플리트는 분석정보의 이해와 모듈화가 가능하므로 재사용을 위한 준비활동과 수행활동을 유기적으로 지원할 수 있다.

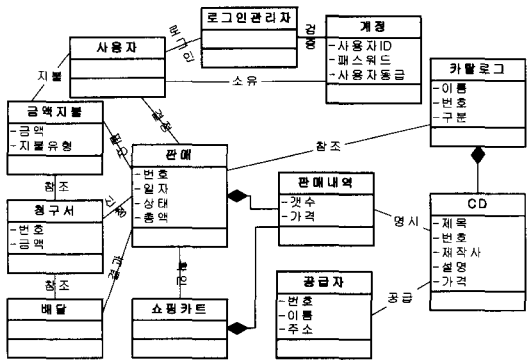
4.3 도메인 템플리트의 적용

본 절에서는 UML을 확장하여 제안한 도메인 템플리트를 이용하여 카탈로그와 쇼핑카트를 지원하는 웹 기반 CD 판매 어플리케이션의 도메인 모델 개발과 재사용을 보인다. 이를 위하여 이전에 개발한 유사 시스템의 산출물을 인터넷 CD 판매 시스템 응용에 적용한다.

도메인 템플리트를 이용한 개발의 첫 활동으로서 새로운 프로젝트 목적 설명서를 검토하고 도메인 템플리트 저장소를 검색하였다. 그 결과 유사한 도메인으로서 물품판매 서브 도메인과 게시판 서브 도메인을 재사용 대상으로 선정하였다. 그림 19-a는 판매 서브 도메인 템플리트로서 사용자가 물품을 구입하는데 있어서 일반적으로 자주 나타나는 상품선택과 금액납부 그리고 배달을 표현한 템플리트이다. 그림 19-b는 게시판 사용서브 도메인 모델 템플리트로서 사용자는 로그인 후에 메시지 제목형태로 나열된 게시판의 사용을



(그림 19) 재사용 도메인 모델 템플리트

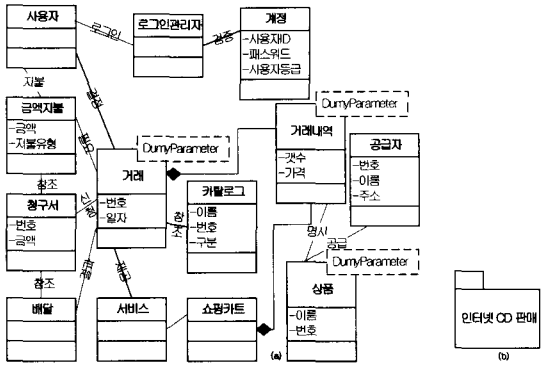


(그림 20) 인터넷 CD 판매 도메인 모델

나타내고 있다.

선택한 템플리트에 있는 도메인 클래스와 템플리트 클래스를 검토하고 인터넷 CD 판매 응용을 위해 인스탄스화, 합성화, 파라미터를 적용하는 일차 템플리트 재사용을 실행하였다. 이어진 일차 도메인 모델 정련화를 통하여 그림 20과 같은 도메인 모델을 얻었다. 그림 19-a의 도메인 클래스 "금액납부"와 파라미터 클래스 "판매물건"은 그림 20에서 도메인 객체 "금액지불"과 "CD"로, 그림 19-b의 클래스 "게시판"과 구체적 항목을 나타내는 "메시지"는 인터넷 CD 판매를 위하여 도메인 객체 "카탈로그"와 이미 선정한 "CD"로 표현되었다. 그리고 특정 항목과 이에 관련된 항목을 다루는 도메인 객체 "쇼핑카트"와 "공급자"가 추가되었다.

인터넷 CD 판매 시스템의 도메인 모델을 개발 후, 다음번 재사용을 위하여 새로운 버전의 도메인인 모델 템플리트를 개발한다. 이전에 개발한 도메인



(그림 21) 새로운 도메인 모델 템플리트

모델에 파라미터 정의, 일반화 및 그룹화를 통하여 인터넷 기반에서 책, 꽃 등 다양한 상품 판매를 다룰 수 있는 인터넷 상품 판매 도메인 모델 템플리트를 그림 21과 같이 개발하였다. 그림 20에 있는 "판매" 및 "판매내역" 도메인 클래스는 "거래", "거래내역" 파라미터 클래스로 재정의 되었으며 도메인 클래스 "CD"는 웹 기반 시스템에서 여러 종류의 상품을 적용할 수 있도록 파라미터 클래스 "상품"으로 표현되었다. 아울러 인터넷 기반에서 쇼핑카트 또는 세금보고서와 같은 상품 판매에 따른 부대 서비스 제공을 위하여 "서비스"라는 이름의 도메인 클래스가 추가되었다. 그림 21-a와 같이 새로 정의된 도메인 템플리트는 그림 21-b처럼 하나의 패키지로 표현하였다.

제안된 표현과 개발 방법의 적용을 통하여 도메인 배경정보를 재사용이 가능한 모델 템플리트 사용요소 단위로 조직화할 수 있었으며 이를 확장된 UML 클래스 다이어그램으로 표현할 수 있었다. 생성된 템플리트에 파라미터, 인스탄스화, 합성화를 적용하여 도메인 모델을 개발할 수 있었다. 아울러, 재사용을 통해 도메인 모델과 도메인 모델 템플리트를 반복 점증적으로 획득할 수 있었다.

5. 결 론

본 논문에서는 객체지향적 시스템의 설계와 객체 모델 생성에 핵심 자료로 사용되는 도메인 모델을

빠르고 경제적으로 개발하기 위하여 UML을 확장한 재사용 표현과 진화적 처리 방법을 연구하였다.

먼저 재사용 가능한 분석정보의 단위적 표현을 위하여 핵심요소, 그룹요소, 사용요소와 같은 구성 요소를 식별하고 수직적 및 수평적 도메인 모델 템플리트의 표현을 지원하도록 UML 클래스와 패키지 다이어그램을 확장하였다. 그리고, 재사용을 위한 표현과 처리의 유기적 작업이 가능하도록 템플리트 정의 활동과 템플리트 재사용 활동을 지원하는 진화적 방법을 제안하였다

제안한 분석정보 재사용을 위한 표현과 개발 방법을 인터넷 CD판매 시스템 개발에 적용하여 UML로 표현되는 도메인 모델을 생성할 수 있었다. 아울러, 점증 반복 작업을 통하여 새로운 재사용 가능한 도메인 템플리트를 획득 할 수 있었다.

본 연구는 우선, 매번 재사용 없이 행해지는 현행 도메인 모델 개발에 재사용을 위한 표현과 처리 방법을 제시하였으며, 주로 코딩과 설계 활동에 국한되어있는 재사용 활동을 분석정보 재사용에까지 확장 가능성을 보였다고 사려된다. 본 논문에서 설계한 UML 기반의 모델 템플리트와 개발 방법은 객체지향적 소프트웨어와 컴포넌트 기반의 소프트웨어 개발을 위한 도메인 모델 템플리트 재사용 도구 개발에 사용될 수 있다.

앞으로의 과제는 UML 기반의 도메인 모델 템플리트 재사용 도구의 설계 및 구현에 관한 연구가 필요하다.

Acknowledgement

본 연구는 2001년도 경원대학교 학술연구비의 지원을 받아 이루어졌음.

참 고 문 헌

[1] C. Thomas, "CORBA Design Pattern", Wiley, pp pxl, 1997.

[2] Craig Larman, "Applying UML and Patterns", 2nd Ed., Prentice Hall, pp. 127, 2001.

[3] D. Rosenberg at al, "Use Case Driven Object Modeling with UML", Addison Wesley, pp. 15~35, 1999.

[4] Alan Brown, "Large-Scale Component-Based Development", Prentice Hall, pp. 105, 2000.

[5] Krzysztof Czarnecki at al, "Generative Programming", Addison Wesley, pp. 61~70, 2000.

[6] Schach, S., "Object Oriented and Classical Software Engineering", 5th Ed., pp12, McGraw-Hill, 2002.

[7] Prieto-Diaz, R., "Domain Analysis for Reusability", Proc. COMPSAC 1987, pp. 23, 1987.

[8] S. Honiden, "객체지향 시스템", 동일출판사, pp. 139, 1996.

[9] R. Ronda, "Software Reuse", Ablex Publishing, pp. 39, 1995.

[10] Martin Fowler, "Analysis Pattern", Addison Wesley, pp. 15, 1997.

[11] Charles Richter, "Design Flexible Object Oriented Systems with UML", Macmillan, pp. 16, 1999.

[12] Grady Booch, "Object Solution", Addison Wesley, pp. 97, 1996.

[13] Ivar Jacobson, "The Unified Software Development Process", Addison Wesley, pp. 121, 1998.

[14] Martin Fowler, "UML Distilled", Addison Wesley, pp. 19, 1999.

[15] Pressman, "Software Engineering", McGraw-Hill, pp. 605, 2001.

[16] Chris Kobryn, "UML 2001", CACM Vol. 42, No 10, pp. 29, 1999.

[17] Grady Booch, "The Unified Modeling Language User Guide", Addison Wesley, pp. 130, 1998.

[18] Mark Priestley, "Practical Object Oriented Design with UML", McGraw-Hill, pp. 264, 2000.

[19] Simon Bennett, "Schaum's Outline of UML", McGraw-Hill, pp. 20, 2001.

◐ 저 자 소 개 ◑

김 지 홍

1982년 미국 California State University (Fullerton) 대학원 전자계산학과 졸업(석사)
1995년 경희대학교 대학원 전자계산공학과 졸업(박사)
1979년~1980년 미국 California State University Computer Laboratory, Consultant
1982년~1989년 미국 Interpac Software Inc., 연구개발 소프트웨어 엔지니어
1989년~현재 : 경원대학교 컴퓨터공학과 부교수
관심분야 : 소프트웨어 공학, UML, 객체지향 분석, 웹 어플리케이션 개발
E-mail : wiskjh@mail.kyungwon.ac.kr