韓國 컴퓨터情報學會 論文誌
第8卷 第4號, 2003. 12.

2003-8-4-1-4

객체모델에 대한 형식명세로의 변환 방법

임 근*, 권영만**

The Translation Method to formal specification of Object Model

Keun Lim*, Young Man Kwon**

요 약

본 논문에서는 정확한 분석 모델을 제시하기 위해서 객체 모델을 정의하고, 이 모델을 정형화와 표준화에 필요한 형식명세로 변환하는 방법을 제안한다. VDM 형식으로 변환된 모델은 정확성, 일관성, 완전성을 제공할 수 있다. 증명의 대상인 VDM 명세에서 오류가 발생한다면 초기 객체 모델 단계에 적용하여 객체 모델의 검증이 가능하다. 검증된 객체 모델을 설계 단계의 기반 명세로 사용하므로 추후 개발 단계의 비용과 노력을 최소화하고 객체 모델 선택의 정확성을 높일 수 있다.

Abstract

In these paper, we define object models in order to represent a correct analysis model, propose translation method to formal specification necessary to uniform and standard. The translated model provide to correctness, consistency and completeness. If it is happen to error in the VDM specification, we can verify model to adapt initial object model step. It increase correctness to retrieval, reduce the costs and efforts of after development because of the verified model used to basic specification in design step.

▸ Keyword : Formal specification, Object model, Model verification

*, ** 서울보건대학 전산정보처리과 조교수

Ⅰ. 서 론

객체 모델링 방법론은 주로 비정형화된 방법을 사용했기 때문에 모호성과 비정확성이 내포된 모델이 생산되기 때문에 모델간 일치성(consistency) 및 완전성(completeness)을 검증할 수 없는 문제점이 있다. 비정형성을 해결하기 위해서는 형식명세 방법이 필요하지만 이것은 수학 이론과 기호에 익숙하지 않은 사용자와의 의사교류를 어렵게 하는 문제점이 있다[1].

따라서 본 논문에서는 객체 모델링 방법인 Shlaer/Mellor의 OOA(Object-Oriented System Analysis) 방법론에서 제공하는 분석 지침과 다이어그램 표기 방법, 그리고 형식 명세 언어인 VDM(Vienna Development Method)에서 제공하는 형식성의 장점을 취하기 위해서 객체 모델을 VDM으로 표현하기 위한 변환 규칙을 제안하였다. 본 논문의 구성은 II장에서는 객체 모델링 방법으로 선택한 Shlaer/Mellor의 OOA, 형식 명세 언어로 선택한 VDM에 대하여 기술한다. III장에서는 객체 모델을 VDM으로 변환하는 규칙들을 제안하고, IV장에서 변환방법의 비교 평가, V장에서는 결론을 기술한다.

Ⅱ. 관련연구

본 논문에서는 객체 모델의 의미와 모델 방법, 그리고 요구에 의해 구성된 각 모델이 재사용을 하기 위해 저장될 수 있는 지의 정확성, 또는 신뢰성의 차원에서 검증이 필요하다고 고려하여 이에 대한 객체 모델에서 형식 명세로의 표현 방법 등을 설명한다.

1. 객체모델링 방법론

객체모델링 방법론의 가장 큰 특징은 다른 모델링 방법론에 비해 구축한 모델들 간의 연결성 및 일관성이 명확하다는 것이다. 정보 모델에서의 정적 구조를 구성하는 각 객체에 대한 동적 특성을 상태 모델에 표현하고, 상태 모델의 각 상태에서 이루어지는 작업을 프로세스 모델에서 표현하고 있다[2].

정보 모델에서는 객체의 속성과 관련성에 관한 정보를 추출할 수 있다. 속성의 경우 기본 키 속성, 참조 속성, 명명 속성, 기술 속성들로 분류하여 표현하므로써 형식 명세의 상태 도메인으로 변형이 자연스럽게 이루어질 수 있다[3][4]. 상태 모델은 상태를 기반으로 하는 형식 명세의 오퍼레이션 추상화를 이루는데 필요한 충분한 정보를 포함하고 있다.

2. 형식 명세 언어

형식 명세 기법은 소프트웨어 요구 정의 및 설계 과정에만 있기 때문에 형식 명세 기법은 소프트웨어 요구정의 및 설계 과정에만 사용되는 것은 아니다. 따라서 명세에서 실제 구현의 과정으로 전이될 때 요구명세에서 발생되는 모호성과 자의적 해석을 피할 수 있다. 형식명세의 접근법은 공리 기반(axiomatic specification)과 구성적 명세로 구분한다[5]. 본 논문에서는 객체 모델의 검증을 위해서 구성적 명세 기법을 이용한다.

이 방법은 집합이론에 근거하여 이미 잘 정의된 모델을 포함하고 있어 새로운 모듈의 구성에 모델을 기반으로 특성을 자료 영역과 특성과 오퍼레이션 특성으로 구분하여 정의한다. 이러한 언어의 예에는 VDM 이나 Z를 들 수 있다. 특히 VDM은 표준화의 노력 및 지원 도구, 특히 검증도구가 풍부하다는 점이다[6]. 따라서 이 방법을 본 논문의 변화대상 명세언어로 선정하였다.

Ⅲ. 변환규칙

본 논문의 연구 범위는 객체지향 환경하에서 재사용 부품을 개발하고, 이들 부품들을 기반으로 부품의 분류 과정을 거치고, 사용자 요구에 부합하는 대상을 검색하여 시스템 구축에 재사용함으로써 시간적 비용적인 측면을 줄이며, 또한 부품의 재사용에 의해서 신뢰도를 증가할 수 있도록 하는데 있다[7].

따라서 본 장에서는 구축할 객체의 정확성 여부를 검증하기 위해서 기존의 소프트웨어 개발 방법론과 형식 명세 방법론을 통합할 수 있도록 변환에 의한 방법을 이용하였다. 객체 지향방법에서 형식 명세언어로의 변환에 의한 방법은 객체 모델을 작성한 후 이를 형식 명세로 전환하는 절차를 따르도록 하기 위해 명세언어에 기반한 변환 규칙을 제시하므로서 개발자가 특정 형식 명세언어에 익숙하지 않아도 검증을 시도할 수 있다는 장점이 있다[7].

3. 객체모델의 변환 방법

객체모델의 작성 과정에서 반영된 객체지향 기본 개념들인 추상화, 상속, 클래스 등을 기반으로, 이러한 개념이 VDM 명세에 최대한 반영하는 것이 기본적인 요구이다. VDM은 개념적으로는 자료추상 영역과 오퍼레이션 추상화로 구분된다. 따라서 이러한 기본적인 개념에 객체지향 방법의 특성을 반영하여 변환한다.

3.1 정적 모델의 변환

정적 모델을 VDM의 상태 도메인으로 변형하기 위해 OOA의 정적 모델에 표현된 각 클래스와 관련성에 대한 VDM의 상태요소를 정의한다. 객체의 정적특성은 객체의 속성과 관련성으로 구성되고 객체의 속성은 그 객체가 갖는 각 특성의 추상화된 표현이다.

(변환방법 S1) 속성

정보모델	VDM
객체	레코드
속성	속성

즉, 정적모델의 객체와 속성은 각각 VDM의 레코드와 이를 구성하는 속성으로 변환한다.

(변환방법 S2) 관련성

정보모델	VDM
관련성(1 : 1)	Map 타입으로 변환
	정의역(객체) 치역(연관관계)
관련성(1 : M)	정의역(제약조건)
	치역(집합)
관련성(M : N)	치역(집합)이 2개인 Map 타입으로 변환
상위/하위 관계	하위타입->레코드타입 상위타입 속성 포함

1 : M의 경우 두개의 Map 타입으로 변환하며 변환될 Map 타입중에 정의역이 M이고 치역이 1인 경우에는 1 : 1의 변환 규칙과 동일하나 도메인이 1이고 치역이 M인 Map 타입은 치역의 객체는 집합으로 정의되고, 정의역에 제약을 가할 때 얻는 치역은 오직 하나가 되도록 변환한다.

(1) 1:1 관련성

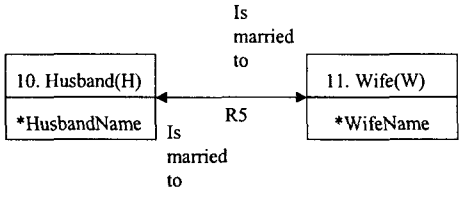


그림 3.1.a 1:1 관련성의 정적 모델
Fig 3.1.a Static model of 1:1 relationship

Husband 객체와 Wife 객체 사이에 정의한 is-married-to 관련성은 Husband의 인스턴스와 Wife의 인스턴스 사이의 양방향 대응성을 모델링하는 것으로 두 개의 Map타입으로 변환한다.

관련성은 객체를 중심으로 연관 객체에 연결된 레이블이 자신과 상대방과의 관련성이므로 아래와 같은 Map 타입의 VDM 명세로 변환된다.

IsMarriedTo1 : Husband $\xrightarrow{m}$ Wife

IsMarriedTo2 : Wife $\xrightarrow{m}$ Husband

inv mk-IsMarriedTo1(is1 : IsMarriedTo1) $\triangleq$
$\forall w \in rng(is1) \cdot \forall h \in dom(is1)\ w = rng(h \lhd is1)$
inv mk-IsMarriedTo2(is2 : IsMarriedTo1) $\triangleq$ $\forall w \in dom(is2) \cdot \forall h \in rng(is2)\ h = rng(w \lhd is2)$

그림 3.1.b 1:1 관련성의 변환 결과
Fig 3.1.b Converted result of 1:1 relationship

M : N 관련성에 관한 의미 정보도 1 : M의 경우와 동일하다. 그러나 M : N의 경우를 정적 모델에서는 관련 객체(associative object)를 따로 두어 모델링하는 것이 바람직하다. 관련 객체는 관련성과 연관된 객체들에 대한 식별자를 갖고 있으며 부가적인 정보를 속성으로 가질 수도 있다.

이런 관련 객체도 하나의 VDM 레코드로 변형되며 이 VDM 레코드는 관련성의 변형인 두 Map 타입과 부가적 정보에 대한 원소들로 이루어진다.

(2) 1 : M 관련성

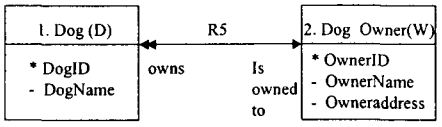


그림 3.2.a 1:M 관련성의 정적모델
Fig 3.2.a Static model of 1:M relationship

(3) M:N 관련성

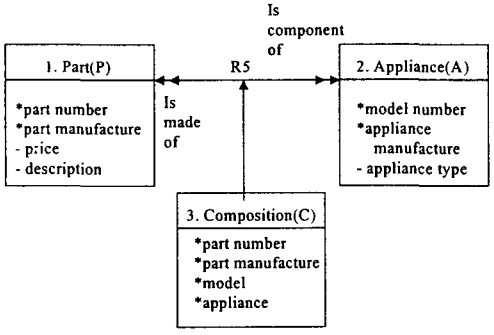


그림 3.3.a M : N의 관련성의 정적 모델
Fig 3.3.a Static model of M:N relationship

이것을 아래와 같은 Map 타입으로 변환할 때, 두 개의 Map 타입이 레코드를 구성하는 속성으로 정의된다.

```
part-set : Set of Part
Appliance-Set : set of Appliance
Composition :: IsMadeOf : Appliance -m- Part-Set
  IsCompositionOf : Part -m- Appliance-Set
inv mk-Part-Set(made : IsMadeOf) △
  ∀parti, part ∈ rng(made)
  dom (made ▷ parti) = dom (made▷partj)
inv mk-appliance-Set(Component : IsCompositionOf) △
  ∀appli, applj ∈ rng(Component)
  dom (Component ▷ appli) = (Component ▷ applj)
```

그림 3.3.b M : N 관련성의 정적 모델 변환 결과
Fig 3.3.b Converted result of M:N relationship

(4) 상위 / 하위 타입 관련성

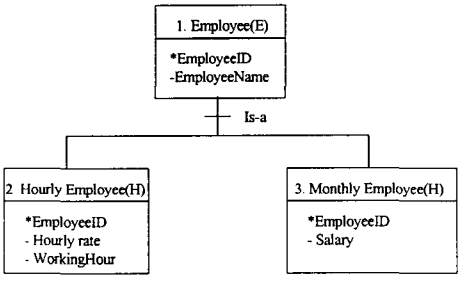


그림 3.4.a 상위/하위 타입의 정적 모델
Fig 3.4.a Converted result of Upper/Lower relationship

〈그림 3.4.a〉는 상위/하위 타입의 관련성을 나타낸다. OOA에서 상위 타입의 인스턴스 생성은 하위 타입의 인스턴스를 생성하지 않고는 불가능하다. 따라서 상위 타입의 객체는 하위 타입의 객체에 의존적이다.

```
HourlyEmployee : EmployeeID
                 EmployeeName
                 Hourly rate
                 WrokingHour

MonthlyEmployee : EmployeeID
                  EmployeeName
                  Salary
```

그림 3.4.b VDM 레코드 타입
Fig 3.4.b VDM record type

1. 동적 모델의 변환

OOA의 상태모델은 한 객체의 행위적 특성을 표현하는 것으로 상태, 상태전이, 행동, 이벤트의 구성요소를 갖고 VDM의 오퍼레이션은 오퍼레이션 시그네처와 오퍼레이션 처리에 필요한 선조건문과 처리후 결과에 대한 후조건문으로 구성된다.

〔변환방법 D1〕 이벤트

동적모델	VDM
이벤트	오퍼레이션 이름
이벤트데이터	매개변수
ObjectID	오퍼레이션 선조건문의 테스트 문장으로 변환

즉, 동적모델의 이벤트 데이터 중 ObjectID는 오퍼레이션의 선조건문에서 해당 인스턴스가 실제로 존재하는지 테스트되는 문장으로 변환된다.

상태모델은 각 이벤트에서 ObjectID를 통해 어떤 인스턴스에게 보내질 이벤트인지 명시하게 된다. ObjectID는 이벤트를 수신하는 인스턴스의 존재 제약성과 이벤트 관련 자료에 관한 제약조건에 대한 기본자료가 된다.

〔변환방법 D2〕 상태

동적모델	VDM
상 태	오퍼레이션 조건영역으로 선조건, 후조건 문장으로 변환

즉, 동적모델의 상태는 오퍼레이션의 조건영역에 해당되며, 전이가 일어나기 전의 상태는 선조건문, 전이가 일어난 후의 상태는 후조건문에서 테스트되는 문장으로 변환된다.

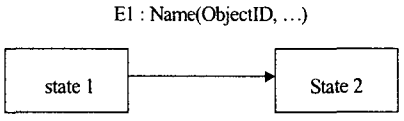


그림 3.5.a 동적 모델의 상태전이
Fig 3.5.a State transition of dynamic model

〈그림 3.5.a〉 과 같은 형태의 동적 모델로부터 기본적인 VDM 오퍼레이션 도메인을 다음과 같이 변환한다.

```
Name(ID:ObjectRecordName, .....)
pre : ID ∈ ObjectRecord ∧ status = state 1
post: status = state 2
```

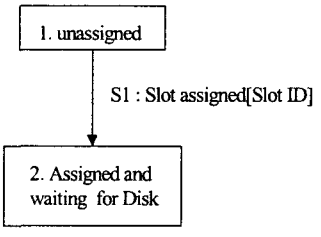
그림 3.5.b 상태전이 변환 모델
Fig 3.5.b State transition of converted model

동적 모델은 정적 모델의 각 객체에 대해 구축하며 각 이벤트에서 ObjectID를 통해 어떤 인스턴스에게 보내질 이벤트인지 명시하게 된다.

〔변환방법 D4〕 행위

동적모델	VDM
매개변수이외 자료수정시	오퍼레이션 변수선언시 ext wr 타입변수로 변환
그 이외 경우	ext rd 타입의 변수로 변환

즉, 동적모델의 행위중에서 매개변수외의 외부자료에 대한 수정이 있을때 오퍼레이션의 변수 선언에서 ext wr 타입의 변수로 변환되며, 그 외의 경우는 ext rd 타입의 변수로 변환된다.



Create Disk TRANSFER with
SourceID := EEPORTID
DestinationID := SlotID
status := "Ready for Entry/Exit Port"
Generate P10 :EEport request pending[EE-PortID]
status := "Assigned and Waiting for DISK

그림 3.6.a DISK TRANSFER 객체의 동적 모델
Fig 3.6.a Dynamic model of Disk Transfer object

〈그림 3.6.a〉의 동적 모델에서 DISKTRANSFER나 EEPORT객체는 모두 외부 요소로 간주될 수 있으며, DISKTRANFER 객체는 그 객체의 자료를 갱신하기 때문에 아래와 같은 VDM 명세로 변환될 수 있다.

```
SlotAssigned(s:Slot, d:DiskTransfer)
ext wr  Disktransfer_Set : Set of Disktransfer
        e : EEPORT
ext rd  Slot_set : Set of Slot
pre  : s ∈ Slot_Set ∧ s.status
       = "unassigned"
post : d ∈ DiskTransfer_Set ∧ d.sourceID
 = e.EeportID
∧ d.DestinationID = s.SlotID
∧ post-EEPortRequestPending(e)
∧ s.status = "Assigning and waiting for DISK"
```

그림 3.6.b 변환된 오퍼레이션 추상화
Fig 3.6.b Converted Operation abstraction

행위 진행 중 다른 객체에게 이벤트를 전송하기도 한다. 이런 이벤트의 전송은 다른 객체의 상태변화를 유발하게 되나, 이런 상태변화 모습을 모두 추적하기는 어렵다. 이 이벤트 처리의 책임은 수신 객체이고 이 수신 객체는 이벤트 처리에 관련된 선조건을 만족함을 보장해야하기 때문이다. VDM으로 변형시에 이벤트 처리의 책임은 수신 객체로 간주하여 현재 변형중인 객체의 VDM 명세에서는 그 이벤트의 후조건이 만족되어야 하는 것만을 고려한다.

VI. 변환방법의 비교

객체 모델을 생성하는 방법은 프로토타이핑 언어를 사용하는 방법과 실행 가능한 형식 명세를 사용하는 방법, Embley가 제안한 CASE인 IPOST를 사용하는 방법이 있다. 본 논문에서는 위의 3가지 방법과 본 논문에서 제안한 방법을 비교 평가 하였다. 평가 결과는 〈표 4.1〉에서 보는 바와 같이, 개발 과정의 단계의 산물인 객체 모델의 사용으로 개발자와 고객간의 일관된 인터페이스를 유지하므로 이해도 증진의 장점을 갖는다.

모델구축환경, 모델 변환필요성, 모델의 이해용이성을 기준으로 비교하였다. 특히 이해 용이성에서는 대부분의 방법에서는 수학적인 기호 또는 프로그램언어를 직접 습득해야 하는 문제점이 있으나 본 논문에서는 다이아그램을 통한 이해를 지원하므로 이해용이성이 뛰어나다.

다음으로 모델 변환필요성에서는 특히 분석단계의 결과를 사용하기 때문에 개발 기간의 단축 및 비용을 절감할 수 있다. 또한 내부 형식 명세 모델을 바탕으로 검증이 가능하다는 장점을 갖는다.

〈표 4.1〉 평가 결과
Table 4.1 Evaluation result

	프로토타이핑 언어	실행가능한 형식 명세	IPOST	본 논문
모델 구축 환경		• 명세 자체가 모델을 표현	• OSA 방법론 지원 • 방법론의 적용 범위가 한정적	• S/M의 OOA 방법론 지원 • 널리 사용되는 방법론 지원 • 타방법론으로의 확장 용이
모델 변환 필요성	• 변환 필요 • 수작업에 의한 변환	• 필요하지 않음 • 논리 언어 사용 • 명세 자체가 실행 가능함	• 필요하지 않음 • 도구 자체의 기본 모델 사용	• 필요하지 않음 • 분석 단계의 결과물인 객체 모델이 직접 적용
이해 용이성	• 프로그래밍 언어 습득 필요	• 수학적인 기호 사용 • 이해하기 어려움	• 다이어그램 이외의 형식 명세 기법의 사용으로 사용자 이해도 저하	• 다이아그램 형태로 제공 • 사용자 관점의 일관성을 위해 명세도구로 다이아그램만 사용

V. 결론

소프트웨어 개발에서 초기 과정인 분석과 설계 단계에서의 오류의 검출은 전체 소프트웨어 개발 노력과 유지보수 비용을 감소시킬 수 있다. 객체 모델링과 형식 명세 기법은 이러한 개발 초기 단계의 부정확한 산출물과 불충분한 모델 구축의 문제를 해결할 수 있는 방법론이다. 객체 모델링은 고객의 요구 사항 추출, 표현의 편리함, 이해 용이성과 같은 장점이 있지만, 비정형화된 방법으로 인하여 모호성과 비정확성을 내포하고 있다. 그러고 형식 명세 기법은 명세에 대한 정확성, 명확성, 간결성을 보장하지만 명세 자체가 수학 이론을 기반으로 하기 때문에 많은 비용을 요구한다.

본 논문에서는 변환방법을 적용하여 상호보완적인 두 방법론의 장점들을 최대화하였다. 이를 위하여 본 논문에서는 객체 모델을 형식 명세 언어인 VDM으로 자동 변환시키기 위한 변환 규칙을 제안하여 변환한 VDM 명세의 객체 지향 모델의 정확성, 일관성, 완전성을 제공할 수 있다. 증명의 대상인 VDM 명세에서 오류가 발생하였다면 원래의 객체 모델에 즉각적으로 반영되어 객체 모델로의 검증까지로 확대된다. 이렇게 검증된 객체 모델을 보다 정확한 설계 단계의 기반 명세로 사용하므로 추후 개발 단계의 정확성을 보장한다.

추후 연구과제로 본 논문은 객체 모델 검증을 위하여 형식명세로 변환하는 방법을 대상으로 하고 있으므로 사용자가 직접 정적모델을 통해서 객체의 관련성 등을 선택하고, 각 정적모델간 상태 또는 이벤트 등의 변환 과정을 직접 제어할 수 있는 시뮬레이션 환경을 구현하는 것이다.

참고문헌

[1] Martin D. Fraser, Kuldeep Kumar, Vijay K. Vaishnavi, "Informal and Formal Requirements Specification Languages: Bridging the Gap," IEEE Transactions on Software Engineering Vol. 17, No. 5, May, 1999, pp. 454-466.

[2] Sally Shlaer and Stephen J. Mellor, Object-Oriented System Anaysis : Object Life Cycles Modeling the World in States, Prentice Hall, 1998

[3] Chris Casey, A Programming Approach to Formal Methods, McGrow-Hill, 1999

[4] 이경환, 소프트웨어 재사용을 위한 객체 모델링 기법, 교학사, 1998

[5] Robert B. Jackson, David W. Embley, Scott N. Woodfield, "Automated Support for the

Development of Formal Object-Oriented Requrements Specification," Proceedings of 6th International Conference on CAiSE'94, 1994

[6] Jeannette M. Wing, "A Specifier's Introduction to Formal Methods," IEEE Computer Vol. 23, No.10, September, 1999

[7] Peter Lindsay, "On transferring VDM verification techniques to Z," Technical Report No. 94-10, Department of Computer Science, University of Queensland

저 자 소 개

임 근

1992. 3 ~ 현재

서울보건대학 전산정보처리과 교수

〈관심분야〉 S/W 공학, 객체지향 방법론, 정보검색 등

권 영 만

1993. 3 ~ 현재

서울보건대학 전산정보처리과 교수

〈관심분야〉 컴퓨터네트웍, 인터넷 응용 등