韓國 컴퓨터情報學會 論文誌
第12卷 第6號, 2007. 12.

2007-12-6-1-14

형식명세로 변환된 객체모델의 검증방법과 시뮬레이션

임 근*

Verification method and Simulation of Object model Converted to Formal Specification

Keun Lim*

요 약

본 논문은 객체모델에서 표현되는 객체와 관련성을 형식명세의 상태와 오퍼레이션 도메인으로의 변환 규칙을 제시하였다. 즉 정보모델의 요소인 객체와 관련성을 형식명세 표현에서 상태영역으로 변환하였다. 동적모델의 상태, 이벤트, 행위를 오퍼레이션 영역으로 변환하였다. 비형식적인 객체모델을 형식 명세 언어로 변환하므로써 객체모델을 정형화된 방법으로 검증할 수 있다. 검증과정을 통해 소프트웨어 개발 초기단계에서 모델링 과정의 편리함과 신뢰성을 제공할 수 있다. 또한 검증된 모델과 사용자 요구사항 사이의 일관성을 위해 시뮬레이션 도구를 구현하였다. 시뮬레이션 도구는 적합한 모델의 선택과 검증이 가능하도록 하므로 소프트웨어 개발 비용과 노력, 개발 시간을 최소화할 수 있다.

Abstract

In this paper, We define convert rules from objects and relation presented in object model to the state and operation domain in formal specification. Namely, object and relation in information model conveted to state domain in formal specification. State, event and behavior converted to operation domain. And that way informal object model change to formal language, it can be verify through formal method. Verification process make an offer convenience and confidence in software development early phase. And we implement simulation tool in order to verification method of formal specification and to consistency verified model between user's requirement. It is possible to select the suitable model and reduce the costs and efforts on software development.

▸ Keyword : SW development, Prototyping language, Model verification

• 제1저자 : 임 근
• 접수일 : 2007.10.31, 심사일 : 2007.12.3, 심사완료일 : 2007.12.15.
* 을지대학교 의료산업학부 부교수

Ⅰ. 서 론

소프트웨어 시스템의 적용 범위가 확장되고 복잡해짐에 따라서 소프트웨어 개발 비용과 유지 보수에 필요한 노력이 증가한다. 따라서 개발 노력과 유지보수 비용을 줄이기 위하여 개발 초기 단계에서 많은 투자가 필요하다. 그 이유는 분석과 설계 단계에서 발견되는 오류가 구현 및 유지 보수 단계에서 발견되는 오류보다 적은 비용으로 수정할 수 있기 때문에 누락시킨 요구사항에 의해 발생하는 유지보수 비용이 크게 감소하기 때문이다[1]. 객체 모델링과 형식 명세 기법(formal specification & methods)은 이러한 개발 초기 단계의 부정확한 산출과 불충분한 모델 구축의 문제를 해결할 수 있는 방법론으로 인식되어 왔다. 사용자의 요구를 정확하게 표현하고, 분석단계에서 발생할 수 있는 오류를 미리 방지하여 고객의 요구를 정확하게 반영할 수 있는 방안을 두 방법론에서 상호 보완적으로 지원하기 때문이다. 그러나 지금까지의 객체 모델링 방법론은 비정형화된 방법을 사용하기 때문에 모호성과 비정확성이 내포되어 모델에 대한 자의적 해석의 가능성이 있다[2]. 이에 비하여 형식 명세 방법은 수학적 이론을 기반으로 잘 정의된 의미를 갖는 기호를 사용하여 소프트웨어 시스템을 모델링 한다[3,4]. 형식 명세는 수학적인 이론을 기반으로 명세하기 때문에, 증명을 통하여 요구사항의 정확성 및 완전성을 보장한다. 그러나 형식명세는 수학적인 이론 및 기호의 사용하기 때문에 요구 명세 작업의 비용을 증가시키며, 이에 익숙하지 않은 고객과의 의사소통을 어렵게 한다[5].

실세계의 자연스러운 표현으로 고객과의 인터페 이스에 적합한 객체 모델링 방법과 증명을 통해 개발과 정확성을 보장하는 형식 명세 방법은 상호 보완적일 수 있다.

본 논문은 형식 명세 언어인 VDM(Vienna Development Method)에서 제공하는 형식성의 장점을 취하기 위해서 객체 모델을 VDM으로 자동 변환하는 접근법을 택하였다. 이를 위해 먼저 변환 규칙을 제안하였다. 이로써 변환된 VDM 명세를 통하여 객체 모델은 검증(verification)을 가능하도록 한다. 이와 함께 검증된 모델을 사용자에게 확인시키기 위한 프로토타이핑 방법으로 모델의 시뮬레이션 방법을 제안한다. 이를 통해서 지금까지의 객체 모델링 방법론이 비정형화된 방법을 사용함으로 인해 모호성과 비정확성이 내포되어 모델에 대한 자의적 해석의 가능성과 정형적인 검증이 곤란했던 단점을 극복하고자 한다.

본 논문의 구성은 II장에서 객체 모델링 방법으로 선택한 Shlaer/Mellor의 OOA, 형식 명세 언어로 선택한 VDM에 대하여 기술한다. III장에서는 형식명세 변환방법의 검증, VI장에서는 검증된 모델을 사용자에게 확인하는 수단으로 시뮬레이션 도구에 대하여 설명하며, V장에서는 결론 및 향후 연구과제를 기술한다.

Ⅱ. 관련연구

본 논문에서는 객체 모델의 정형화된 검증을 위해 변환 기법을 적용하였다. 이를 위해 변환 대상인 객체 모델은 Shlaer/Mellor의 OOA 방법론을 택하였고, 결과로 얻어질 형식 명세 언어는 VDM을 택하였다. 이 장에서는 각각의 특성을 기술한다.

2.1 객체지향 분석 방법론

Shlaer/Mellor 방법론은 산업계에서 소프트웨어 개발에 적용할 수 있는 잘 정의된 체계적 접근법으로 평가 받고 있다. 방법론의 가장 중요한 특징은 대형 프로젝트를 도메인으로 분할한 후 각 도메인별로 분석 과정을 진행하므로써 복잡도 관리의 효과적 방법을 제공한다. Shlaer/Mellor의 객체지향분석(Object Oriented Analysis)를 통해 어플리케이션 도메인(Application Domain)을 분석하는 것이 다음 단계이다. 이 단계에서는 도메인의 객체와 객체들 사이의 관련성을 표현하는 정보 모델(Information Model), 각 객체의 행위와 객체들 간의 상호 작용 관계를 보여주는 상태 모델(State Model), 상태 모델이 요구하는 처리 과정을 보여주는 행위 자료 흐름 다이어그램(Action Data Flow Diagram : ADFD)을 만들게 된다[2].

2.2 정보모델

정보 모델은 문제의 기본 객체와 그들간의 관련성을 추출하여 표현하므로써 어플리케이션의 구조적 모습을 보여준다. 그러므로 정보 모델의 궁극적 목적은 문제 영역에 속한 객체를 추출하여 표현하는 것이다[6]. 정보 모델은 객체와 이를 구성하는 속성들과 객체들간의 관련성을 표현하는 정보 구조 다이어그램(Information Structure Diagram)으로 표현한다. 이와 함께 정보 구조 다이어그램에 포함된 각 정보에 대한 기술서로 객체와 속성 기술서와 관련성 다이어그램을 제공한다. OOA 방법론에서의 객체는 다른 방법론에서와 같이, 동일한 속성을 지닌 실세계 대상들을 추상화한 것으로

정의하고, 속성들은 모든 인스턴스(Instance)가 지니고 있는 자료의 추상화로 간주한다. 객체들이 하나의 어플리케이션에 포함될 때는 일정한 패턴으로 상호간의 관련성를 갖는다. OOA 방법론이 제공하는 관련성(Relationship)은 일반적 관련성과 타입간의 개념적 관계를 표현하는 상위/하위 타입(Supertype/Subtype)의 관련성이다. 객체와 관련성의 식별이 완료되면 동적 모델링 단계에서 각 객체에 대한 상태와 상태 전이의 개념으로 객체의 동적 특성을 정형화하는데, 상태 모델은 정보 모델을 구성하는 각 객체에 대해 모델링한다. 상태는 객체가 가질 수 있는 물리적, 논리적 특징과 법칙을 추상화한 것으로 다른 상태의 행위적 관점에서의 객체를 표현한다[7].

2.3 정형화 표현

OOA 방법론의 가장 큰 특징은 다른 모델링 방법론에 비해 구축한 모델들간의 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. 정보 모델에서는 객체의 속성과 관련성에 관한 정보를 추출할 수 있다. 속성의 경우 기본 키 속성, 참조 속성, 명명 속성, 기술 속성들으로 분류하여 표현하므로써 형식 명세의 상태 도메인으로 변형이 자연스럽게 이루어질 수 있다[8]. 상태 모델은 상태를 기반으로 하는 형식 명세의 오퍼레이션 추상화를 이루는데 필요한 정보를 포함하고 있다[9]. 단, 각 상태를 구성하는 행동들을 기술하는 정형화된 방법이 존재하지 않으므로 모델을 구성하는 사람의 의도를 자동으로 추출하기 어렵지만, 정형화된 구문과 의미를 제공한다면 해결할 수 있다[10]. 프로세스 모델은 상태 모델의 각 상태에서 이루어지는 작업을 정형화한 다이어그램의 형태로 표현하고자 하는 의도를 지니고 있으나, 객체 저장소와 객체의 표기법은 형식화되었으나, 프로세스 자체는 상태 모델과 동일한 수준이다. 그러나 이 프로세스 모델을 오퍼레이션 추상화가 함수적 추상화(functional abstraction)인지 순수 오퍼레이션에 의한 추상화(operational abstraction)인지를 구분할 수 있는 중요한 단서를 제공한다. Shlaer/Mellor의 세가지 모델은 VDM과 같은 모델 기반 형식 언어와 연계성이 가장 뛰어난 것으로 판단된다.

III. 형식명세 변환방법의 검증

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 증가시키는데 있다. 이를 위해서 재사용 대상이 되는 객체를 모델링 단계에서부터 모델간 일치성과 완전성을 검증하여 가장 바람직한 대상을 재사용 라이브러리에 저장하고 여기에서 시스템 개발시 필요로하는 부품을 검색할 수 있도록 지원하는 프로토타입 시스템 개발에 있다. 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 장점을 갖는다.

3.1 객체모델의 변환 과정

지금까지 소프트웨어 개발 방법론과 형식 명세방법은 서로 통합되지 못하고 병행적으로 접목되었다. 즉, 임의의 소프트웨어 개발방법론에 의해 소프트웨어를 개발하고 이와 병행하여 형식이론을 도입한 소프트웨어의 개발이 이루어졌다. 이러한 방식은 두 흐름을 동시에 존재시켜 개발하므로써, 보다 정확하고 안정적인 시스템을 얻을 수 있다. 그러나 형식이론에 의한 개발은 수학적 이론에 익숙한 개발자만이 가능하기 때문에 서로 다른 개발의 흐름을 병행하기 보다는 두 방법의 통합을 이루는 것이 바람직하다고 본다. 본 논문에서는 객체모델링 방법인 OOA를 대상으로 하고 목적명세는 집합이론에 근거한 VDM을 대상으로 선정하여 두 방법의 통합을 변환의 과정을 통해 시도하고자 한다. 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. 〈그림1〉과 같이 변환과정은 정적 모델을 상태 도메인으로 변환하는 과정과, 동적모델을 오퍼레이션 도메인으로 변환하는 두 개 과정으로 구분하였다.

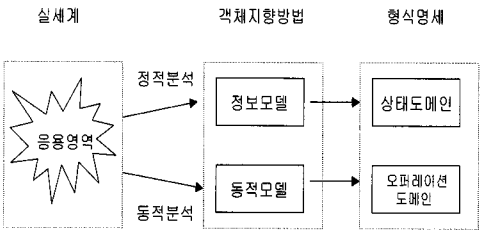

그림 1. 객체모델의 변환 과정
Fig 1. Convert Process of Object Model

3.2 변환 방법의 검증

본 장에서는 객체 모델의 정확성 검증을 위해 변형 방법의 정의를 통해 이미 검증 이론이 정의된 형식명세 언어인 VDM을 이용한 명세를 생성하였다. 변환된 VDM 명세의 정확성을 보장하기 위해서는 정의한 변환방법의 검증이 필요하다. 변환방법의 검증을 위해서 정확성(Correctness), 완전성(Completeness)을 확인해야 한다.

3.2.1 완전성

객체모델의 요소 모두가 VDM의 구성요소로 변환됨을 검증하는 것이고, OOA의 정적모델과 동적모델의 각 구성요소와 VDM간의 대응성은 〈표 1〉과 같이 정리할 수 있다.

3.2.2 정확성

정확성을 확인하기 위해서는 변환 이전의 객체 모델이 갖고 있는 정보와 변환 방법의 적용으로 생성한 VDM 명세가 표현하는 모델 사이의 정보 손실 또는 추가가 없음을 보장해야 한다. 객체는 자료와 행위적 특성 모두를 포함하는 단위이지만 VDM은 자료 도메인과 행위 도메인이 분리 기술되기 때문에, VDM 하나의 구성요소로 객체 모델의 전체를 표현할 수 있다. 따라서 자료 도메인과 행위 도메인을 구분하여 정보의 첨삭 여부를 확인한다.

표 1. 객체 모델과 VDM간 대응관계
Table 1. Correspond Relation Object to VDM

객체모델		VDM
정적모델	객체	Record Type, Set Type
	속성	Fields of Record
	관련성	Map Type
	상속성	Record의 중복
동적모델	상태	Pre or Post
	이벤트	Operation or Function
	전이	Pre, Post의 명세로 상태 변화 표현
	행동	Post를 구성하는 논리 연산문

자료적 측면의 검증은 데이터 값들의 보존 여부로 확인할 수 있다. 〈그림 2〉의 객체 모델은 〈그림 3〉의 정보를 구성하게 된다. 〈그림 2〉는 정적모델의 한가지 예로서 다대다의 관련성을 갖는 모델의 표현이며 이것을 구체적인 정적모델에서 표현하는 정보로 보여주는 과정이 〈그림 3〉이다.

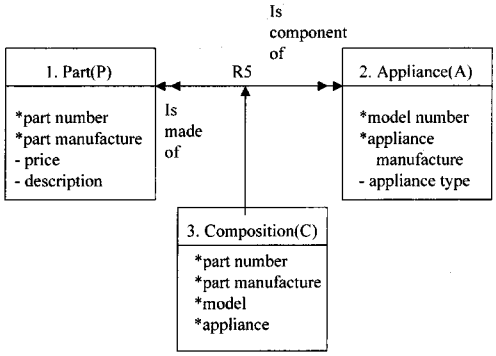

그림 2. 다대다 관련성의 정적모델
Fig 2. Static Model of M:N Relation

(a) Part(P)

Part Number	Manufacture	Price	Description
103	GE	$100	
21	Seara	$523	Not in stock
986	Radio Shack	$345	

(b) Application(A)

Model Number	Manufacture	Type
176A	GE	Washer
92AB	Maytag	Dryer
12345	Sears	Cleaner

(c) Composition(C)

Part Number	Part Manufacture	Model Number	Appliance Manufacture
103	GE	176A	GE
103	GE	92AB	Maytag
21	Seara	12345	Sears
986	Radio Shack	12345	Sears

그림 3. 정적 모델의 정보 예
Fig 3. Information Example of Static Model

객체모델을 변형하여 얻어진 〈그림 4〉의 VDM의 상태 도메인에서 얻어지는 정보는 다음과 같다.

part-set : Set of Part
Application-Set : set of Appliance

Composition :: IsMadeOf : Appliance $\overset{m}{-}$ Part-Set

IsCompositionOf : Part $\overset{m}{-}$ Appliance-Set

inv mk-Part-Set(made : IsMadeOf) $\triangleq$
$\forall part_i, part \in rng(made)$
$dom\ (made \rhd part_i) = dom\ (made \rhd part_j)$

inv mk-appliance-Set(Component : IsCompositionOf)
$\triangleq \forall appl_i, appl_j \in rng(Component)$ dom (Component $\rhd appl_i$) = (Component $\rhd appl_j$)

그림 4. 정적 모델이 VDM으로 변환된 결과
Fig 4. Converted Result Static Model to VDM

변형된 VDM에서 얻어진 정보는 변형 이전의 객체 모델이 갖는 정보와 동일한 내용을 표현하고 있음을 보여주고 있다. 〈그림 5〉 정적모델에서 추출된 객체, 속성 값이 추출된 결과이다. 이로써 정적 모델의 변형 방법의 정확성을 검증할 수 있다.

Part = { (103), "GE", $100, " " ,(21, "Sears", $523, "Not In Stock"), (986, "Radio Shack", $345, " " }
Application = { (176A, "GE", "Washer") , (92AB, "Maytag", "Dryer"), 12345, "Sears", "Cleaner")}
Composition = {((103, "GE",) -> {(176A, "GE"), (92AB, "Maytag")}), ((21, "Sears")-> {(12345, "Sears")}), ((986, "Radio Shack") -> {(12345, "Sears")}), (12345, "Sears") -> {(986, "Radio Shack"), (21, "Sears")}), (((176A, "GE"), -> {((103, "GE",)}),((92AB, "Maytag") -> (103, "GE",)})}

그림 5. VDM에서 추출된 결과
Fig 5. Extracted Result from VDM

K1 : Item order made[customer ID, catalog number]
1. Entered
K3:Back order customer ID, [customer ID, catalog number, item number]
Create Item Order witn catalog and new number
If Catalog Item Quantity On Hand -> o then
Decrease quantity on hand
Generate K2: Fulfill order [customer ID, catalog number, item number]
else
Generate K3: Back order[customer ID, catalog number, item number]

Current state :="entered"
2. Back ordered
Tell the customer that the item is a back order
Order the catalog item from the manufacturer
Current state : = "back ordered"
K4:Supply received from manufactureer[customer ID, catalog number, item number]
K2: Fulfill order [customer ID, catalog number, item number]
3. Item Shipped
Pack and ship items with invoice
Current state: ="item shipped"
K5: Payment received [customer ID, catalog number, item number]
4. Paid
Delete item order

그림 6. Item Order에 대한 동적 모델
Fig 6. Dynamic Model of Item Order

Customer ID, Catalog Number, item number, current state의 네가지 속성을 정의하고 있는 Item객체의 동적모델은 〈그림 6〉과 같다. ItemOrder와 연관된 객체 Catalog의 상태를 가정한 단계를 거치기는 하였으나 이는 현재 동작중인 다른 객체의 상태 테이블의 파악으로 간주될 수 있다.

(a) K1 : ItemOrderMade(100,120) 이벤트 발생 이전의 객체 상태 테이블

Customer ID	Catalog Number	Item Number	Current State
240	RES123	234	Entered
124	BOOK14	111	Entered
688	CD789	567	Back Ordered
135	RECORD12	135	Item Shiped

(b) K1 : ItemOrderMade(100,120) 이벤트 발생 이후의 객체 상태 테이블

Customer ID	Catalog Number	Item Number	Current State
240	RES123	234	Entered
124	BOOK14	111	Entered
688	CD789	567	Back Ordered
135	RECORD12	135	Item Shiped
100	120	145	Entered

(c) K1 : ItemOrderMade(100,120) 이벤트 처리 과정에서 발생하는 K2 객체 상태 테이블

Customer ID	Catalog Number	Item Number	Current State
240	RES123	234	Entered
124	BOOK14	111	Entered
688	CD789	567	Back Ordered
135	RECORD12	135	Item Shiped
100	120	145	Item Shiped

그림 7. 객체 Item Order의 상태변화 과정
Fig 7. State stransition Step of Item Order

변환한 VDM명세의 일부 오퍼레이션 명세를 기반으로 상태 도메인을 확인하면 〈그림 7〉과 동일한 결과를 얻게 되어 동적 모델의 변형 방법 역시 완전함을 확인할 수 있다. 〈그림 8〉은 〈표 1〉에서 정의한 동적모델의 상태, 이벤트, 전이, 행동 등에 관한 정보가 형식명세언어로 변환된 결과를 보이고 있다. 변환이전과 이후의 정보의 내용이 일치함을 확인할 수 있다. 따라서 변환 결과의 완전성이 검증된다.

```
ItemOrderMader (c : Customer , cn : Catalog Number , i : ItemOrder)
    ext rd: CatalogItem : ItemofCatalog
    pre : i ∈ ItemofCatalog
post : i ∈ ItemOrder-set and i.CustomerID = c.CustemerID
                and
    i.CatalogNumber = c.CalogNumber and i.itemNumber
                = rand()
        and if CatalogItem.QuantityOnhnd 〉 0
        then post-FulfullOrder(c,cu, i.itemNumber)
        else post-Backorder(c,cu,i,itemNumber)
        and i.Status = "Entered"
```

그림 8. 동적모델이 VDM으로 변환된 결과
Fig 8. Converted Result Dynamictatic Model to VDM

IV. 시뮬레이션에 의한 모델의 검증

본 논문에서는 형식 명세를 기반으로 객체 지향 분석 모델의 검증을 가능하게 하고, 이 모델과 사용자의 요구사항 간의 확인을 위해 시뮬레이션 환경을 구축하였다. 이 환경에 의해 생성되는 각 프로토타입은 시스템이 갖추어야 할 기능성들은 설명문구나 단순한 외적 모형보다는 개발자와 사용자간의 의사 소통상의 효과를 증진시킬 수 있는 동적 시각 모형을 제공함으로서 단순한 인터페이스 수준의 프로토타입이 갖는 제한성을 해결할 수 있다.

4.1 시뮬레이션 도구의 설계

시뮬레이션 도구는 객체 모델링의 결과물인 정보 모델과 동적 모델을 참조하여 각 객체와 시스템 내의 상호 작용을 보여주며 이러한 역할을 위하여 시뮬레이션 도구는 〈그림 9〉와 같이 객체 모델 편집기, 뷰어, 시뮬레이션 처리기로 구성하였다.

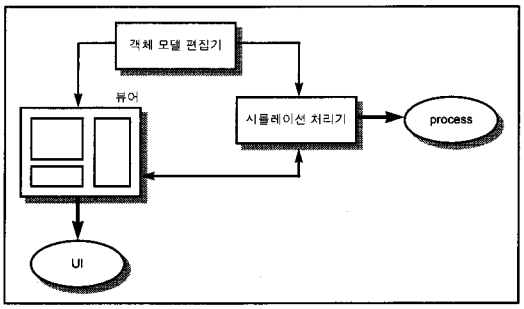

그림 9. 시뮬레이션 도구
Fig 9. Simulation Tool

(1) 객체 모델 편집기

객체 모델 편집기는 사용자가 쉽게 객체 모델을 작성할 수 있도록 다이어그램을 제공한다. 편집기에서 제공하는 다이어그램은 OOA의 정보 모델에서 필요로 하는 요소와 동적 모델에서 필요로 하는 요소로 구성된다. 정보 모델에서 필요한 요소는 객체, 관련성, is-a 관련성이며, 동적 모델에서 필요한 요소는 상태, 상태 전이, 이벤트, 행동등이다. 이러한 요소를 제공하여 사용자가 쉽게 모델링할 수 있는 기능을 제공하며 모델 자체를 시뮬레이션 도구의 입력으로 사용될 수 있는 형태로 저장했다.

(2) 뷰어

뷰어는 시뮬레이션 진행 상황을 보여주는 것으로 3개의 화면으로 분할하여 다양한 관점에서의 뷰를 제공한다. 〈그림 10〉〈그림 11〉이 뷰어에 해당한다.

① 시뮬레이션 진행 화면
② 인스턴스의 상태 머신 화면
③ 동작화면윈도우

(3) 시뮬레이션 처리기

시뮬레이션을 관리하고 사용자와의 상호 작용을 처리하는 부분으로 이벤트 리스트를 통하여 시뮬레이션의 스케쥴을 결정하고, 실제로 이벤트가 도달할 객체에게 메세지를

전달하여 전이를 일으켜 실제 시뮬레이션을 담당하여 그 결과를 뷰어를 통하여 보여준다.

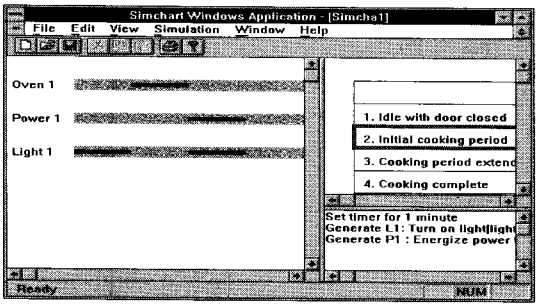

그림 10. 시뮬레이션 화면
Fig 10. Simulation View

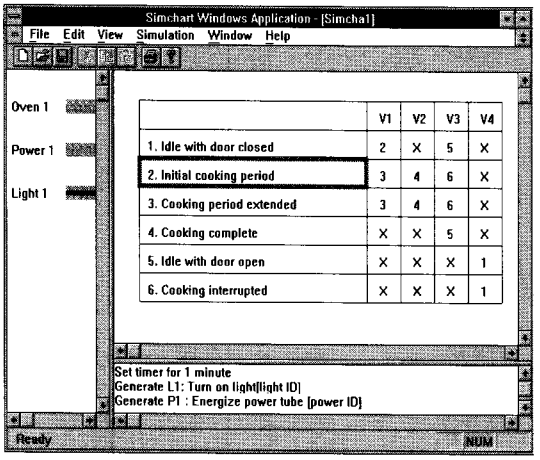

그림 11. Oven 객체의 상태 전이과정
Fig 11. State Transition of Oven Object

4.2 시뮬레이션 도구의 구현

〈그림 10〉은 Oven, Power Tube, 그리고 Light로 구성된 정보 모델을 객체 모델 편집기를 이용하여 그리는 화면이다. 〈그림 10〉은 시뮬레이션 도구의 실행 화면으로 Oven, Power Tube, Light의 객체를 각각 1개씩 인스턴스화하여 과정을 보이고 있다. 즉 오븐의 구성 및 처리과정을 정적모델과 동적모델로 구분하고, 정적모델 요소인 객체 oven, power, tube간 관련성을 시각적으로 보여주고 있으며, 〈그림 11〉에서 각 객체의 동적모델 요소인 상태 및 이벤트, 전이 과정을 확인하면서 모델의 변환 및 선택된 객체의 검증을 쉽고, 용이하게 진행할 수 있도록 한다. 즉 오븐의 상태 변화과정과 각 객체와의 연관성을 Vn으로 표현하고 있다. 시뮬레이션 도구를 이용하여 사용자의 요구사항을 수렴하고 정확성을 검증함으로서 바람직한 모델선택 기준을 제시할 수 있다. 특히 개발과정에서 정형화된 객체모델을 사용하기 때문에 개발자와 고객간 일관된 인터페이스를 유지할 수 있고 모델에 대한 이해도를 증진시킬수 있는 장점을 갖는다.

V. 결론 및 향후 연구과제

본 논문에서는 정보 모델에서 표현되는 객체와 관련성을 형식명세로 표현하는 데이터 구조를 이용하여 VDM의 상태 영역으로 변환하였고, 동적 모델의 상태, 이벤트, 이벤트 데이타, 행위를 VDM의 오퍼레이션 영역으로 변환하였다. 이를 통해 소프트웨어 개발단계인 초기 과정에서 분석가에게 모델링의 편리함과 자율적인 재량을 제공함과 동시에 모델에 대한 정형적인 검증 방법을 제공한다는 장점을 얻을 수 있다. 그리고 비형식적인 객체 모델을 형식 명세 언어인 VDM으로 변환하므로써 객체 모델을 정형화된 방법으로 검증할 수 있는 수단을 제공한다. 변환된 VDM 명세는 이미 연구된 검증 방법론 및 도구의 지원으로 정확성, 완전성을 검증하게 된다. 변환 과정과 검증 과정사이에서 발견한 논리적 오류를 통해 객체 모델에 포함된 오류를 식별하고 수정한다. 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 사용하므로써 추후 개발 단계의 정확성을 보장한다. 이와 함께 고객의 요구사항에 대한 확인 작업은 이미 검증된 객체 모델을 시뮬레이션 도구를 이요하여 사용자의 요구사항에 부합하는 모델을 선택할 수 있도록 하였다. 시뮬레이션은 객체 모델링의 결과물인 정보 모델과 동적 모델을 참조하여 시스템내의 각 객체간의 상호 작용을 동적인 시각 모형으로 제공하여, 개발자들과 사용자들간의 의사 소통상의 효과를 크게 증진시킬 수 있다.

추후 연구과제로는 본 논문의 적용범위가 확대될 경우 모든 개발 모델을 형식명세로 변환하면서 발생할 수 있는 복잡성 문제를 해결해야 한다.

이를 해결하개 위해서 모델의 자동화분류 및 검색에 관한 연구가 필요할 것이다.

참고문헌

[1] 이경환, 소프트웨어 재사용을 위한 객체 모델링 기법, 교학사, 1993

[2] 임근, 권영만 , 객체모델에 대한 형식명세로의변환 방법, 한국컴퓨터정보학회논문지,제8권 4호 2003년 12월

[3] Sally Shlaer and Stephen J. Mellor, Object-Oriented System Anaysis : Object Life Cycles Modeling the World in States, Prentice Hall, 1992

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

[5] 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.

[6] 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

[7] Jeannette M. Wing, "A Specifier's Introduction to Formal Methods," IEEE Computer Vol. 23, No.10, September, 1990

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

[9] Charlie Alfred and Stephen J. Mellor, Observation on the Role of Patterns in object-oriented sw development, Object Magazine, 61-65, 95

[10] D. J Chen, "A Model driven apperoach to accessing managerial information: The development of a repository-based executive information system" J.Management Information system, vol 11, No 4, 33-66, spring 95.

저 자 소 개

임 근

1998년 중앙대학교 컴퓨터공학과 공학박사

1992년 - 현재 을지대학교 의료산업학부 부교수

관심분야 : 정보검색, 데이터마이닝, 재사용 방법