논문 2009-5-24

객체 모델 선택을 위한 검증 및 검색방법

The Verification and Retrieval Method for selection of Compatible Object Model

임명재*, 권영만**, 강정진***

Myung-Jae Lim, Young-Man Kwon, Jeong-Jin Kang

요 약 본 논문에서는 정확한 분석 모델을 제시할 수 있는 객체 모델링과 정형화 방법을 통해 개발자와 사용자간 효과적인 의사소통을 제공하고 객체모델의 정형화와 표준화에 필요한 형식명세로의 변환 규칙을 제안한다. 사용자의 요구에 따라 최적의 객체모델 선택을 위한 객체 모델 검색 프로토타입을 제시한다. 이를 통해서 적합한 모델을 선택할 수 있으므로 소프트웨어 개발시 비용과 노력을 최소화할 수 있다

Abstract In this paper, We define convert rules objects and relation presented in object model to the state and operation domain in formal specification. 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.

Key Words : Object Modeling, Formal specification, Verification Method, Convert Rule

Ⅰ. 서 론

소프트웨어 시스템의 적용 범위가 확장되고 복잡해짐에 따라서 소프트웨어 개발 비용과 유지 보수에 필요한 노력의 증가가 요구되며, 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 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. 그러나 지금까지의 객체 모델링 방법론은 비정형화된 방법을 사용하기 때문에 모호성과 비정확성이 내포되어 모델에 대한 자의적 해석의 가능성이 있다. 비정형성으로 인하여 모델간의 일치성(consistency) 및 완전성(completeness)을 검증할 수 없다는 단점을 갖는다[2,3]. 이에 비하여 형식 명세 방법은 수학적 이론을 기반으로 잘 정의된 의미를 갖는 기호를 사용하여 소프트웨어 시스템을 모델링 한다[3,4]. 실세계의 자연스러운 표현으로 고객과의 인터페이스에 적합한 객체 모델링 방법과 증명을 통해 개발과 정확성을 보장하는 형식 명세 방법은 상호 보완적일 수 있다.

따라서 본 논문에서는 객체 모델링 방법에서는 고객의 요구사항 추출, 요구사항 표현의 편리함, 의사교류, 이해 용이성, 설계의 기반 자료의 확보라는 장점과 형식 명세 방법에서는 정확성, 명확성, 그리고 간결성의 장점을 통해 시스템의 분석과 검증에 적용할 수 있다.

*종신회원, 을지대학교 의료산업학부
**정회원 을지대학교 의료산업학부(교신저자)
***종신회원, 동서울대학 정보통신과
접수일자 2009.9.5, 수정일자 2009.10.6

본 논문의 구성은 Ⅱ장에서는 관련연구, Ⅲ장에서는 객체 모델 검증 및 검색 프로토타입을 설명하고, Ⅳ장에서는 비교평가 Ⅴ장에서 결론을 기술한다.

Ⅱ. 관련연구

1. 객체지향 분석방법

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 OOA를 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. 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[5,6]. 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 포함될때는 일정한 패턴으로 상호간의 관련성를 갖는다. OOA 방법론이 제공하는 관련성(Relationship)은 일반적 관련성과 타입간의 개념적 관계를 표현하는 상위/하위 타입(Supertype/Subtype)의 관련성이다.

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. 상태 모델은 상태를 기반으로 하는 형식 명세의 오퍼레이션 추상화를 이루는데 필요한 충분한 정보를 포함하고 있다. 단, 각 상태를 구성하는 행동들을 기술하는 정형화된 방법이 존재하지 않으므로 모델을 구성하는 사람의 의도를 자동으로 추출하기 어렵지만, 정형화된 구문과 의미를 제공한다면 해결할 수 있다.

3. 형식명세언어

프로세스 모델은 물론 상태 모델의 각 상태에서 이루어지는 작업을 정형화한 다이어그램의 형태로 표현하고자 하는 의도를 지니고 있으나, 객체 저장소와 객체의 표기법은 형식화 되었으나, 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[6].

Ⅲ. 객체 모델의 검증 및 검색 프로토타입

프로토타입은 향후 개발하고자 하는 시스템의 본질적 특성만을 보여주는 개발 초기의 모델이다. 본 논문에서는 형식 명세를 기반으로 객체 지향 분석 모델의 검증을 가능하게 하고, 이 모델과 사용자의 요구사항간의 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 프로

토타입이 갖는 제한성을 해결할 수 있다.[7]

1. 시뮬레이션 도구의 설계

시뮬레이션 도구는 객체 모델링의 결과물인 정보 모델과 동적 모델을 참조하여 각 객체와 시스템 내의 상호작용을 보여주며 이러한 역할을 위하여 시뮬레이션 도구는 크게 객체 모델 편집기, 뷰어, 시뮬레이션 처리기로 구성하였다. 각 도메인과 도메인간의 연결성은 <표 1>과 같다.

표 1. 도메인과 도메인과의 연결성
Table. 1. Domain and Relation with Domain

도메인 및 연결성	설명
시뮬레이션 도구	정보 모델과 동적 모델로부터 시뮬레이션 정보를 추출하여 시뮬레이션 과정을 제어한다.
사용자 인터페이스(UI)	시뮬레이션의 결과를 처리하며 사용자의 부가 정보 입력을 처리한다.
프로세스(Process)	시뮬레이션 과정에 생성하게 되는 각 객체의 처리 행위 동적 모델에 내포된
타이머(Timer)	Timer 객체로 시뮬레이션에 필요한 시간적 제약성 처리
구조(Architecture)	시뮬레이션 설계에 필요한 객체들의 집합으 로 C++ 언어, 각종 자료 구조 객체들의 집합
시뮬레이션 도구 ⇒ UI	시뮬레이션 과정 중에 생성한 객체의 상태에 관한 정보를 제공한다.
UI ⇒ 시뮬레이션 도구	시뮬레이션에 필요한 부가 정보를 사용자로 부터 입력받아 제공
타이며 ⇒ 프로세스	프로세스 수행 또는 지연 시간 제약

시뮬레이션 지원 도구의 가장 중요한 도메인은 시뮬레이션 도구로 이를 서브 시스템으로 분할하면 <그림 1>과 같다.

2. 모델검증 및 규칙

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 위해서는 정확성(Correctness), 완전성(Completeness)을 확인해야 한다.

(1) 완전성

객체모델의 요소 모두가 VDM의 구성요소로 변환됨을 검증하는 것이고, OOA의 정적모델과 동적모델의 각 구성요소와 VDM간의 대응으로 정리할 수 있다[8].

(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 보장해야한다. 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[9].

(3) 모델 구성 규칙

정보 모델을 구축하는 규칙을 정의하면 다음과 같다.

1) 모든 객체는 속성들 중 적어도 하나의 식별자를 갖고 이를 * 기호를 통해 구별한다.
2) 관련성은 <관련성 이름>의 참조 속성으로 모델링한다.
3) 각 관련성은 관련성의 종류와 관련성의 수(Cardinality)를 표기한다. 관련성의 수로는 일대일, 일대다, 다대다를 표현한다.
4) 관련성으로 인해 유도되는 속성이 존재한다면 이를 연관 객체(Associative Object)로 분리하여 모델링한다.
5) Supertype/Subtype 관련성에 의해서는 상위 객체의 속성을 하위 객체가 모두 상속받는 관계를 모델링한다.

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.

3. 시뮬레이션 시스템 구성요소

(1) 객체 모델 편집기

<그림 1>에서와 같이 객체 모델 편집기는 사용자가 쉽게 객체 모델을 작성할 수 있도록 다이어그램을 제공

한다. 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, 행동 등이다. 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 화면에 3개의 윈도우로 분할하여 다양한 관점에서의 뷰를 제공한다.

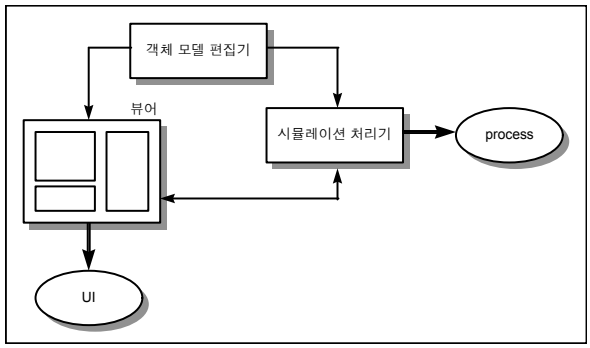

그림 1. 시뮬레이션 시스템 구성도
Fig. 1. Simulation system Diagram

① 시뮬레이션 진행 윈도우
② 인스턴스의 상태 머신 윈도우
③ 행동윈도우

(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.

4. 시뮬레이션 도구의 구현

<그림 2>는 Oven, Power Tube, 그리고 Light로 구성된 정보 모델을 객체 모델 편집기를 이용하여 그리는 화면이다.

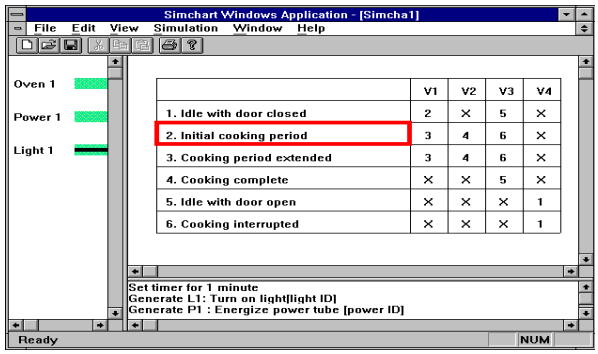

그림 2. Oven 객체의 상태 전이표
Fig. 2. Oven Object State-transition table

<그림 2>는 시뮬레이션 도구의 실행 화면으로 Oven, Power Tube, Light의 객체를 각각 1개씩 인스턴스화하여 시뮬레이션한다. 그림에서 가로축은 이벤트의 처리 순서를 의미한다. 먼저 Light 객체에 이벤트가 도달하고 그리고 Oven 객체, 다음은 Power Tube와 Light 객체에 동시에 이벤트가 도달함을 의미한다.

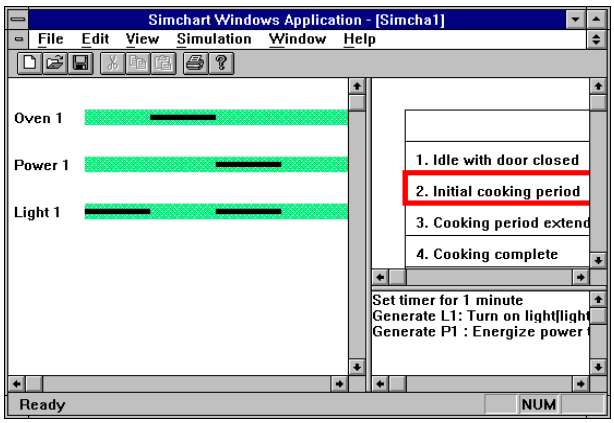

그림 3. 시뮬레이션 화면
Fig. 3. Simulation View

<그림 3>은 각 객체의 상태를 상태 전이표를 이용하여 보여준다. Light 객체가 먼저 동작하고, 이어 Power Tube의 불이 들어오면 이어 선택한 Oven 객체의 기능이 진행되는 과정을 보여주고 있다.

IV. 비교평가

객체 모델의 사용자 확인을 위하여 프로토타입을 생성하는 방법은 프로토타이핑 언어를 사용하는 방법과 실

행 가능한 형식 명세를 사용하는 방법, Embley가 제안한 CASE인 IPOST를 사용하는 방법이 있다.

표 2. 비교평가
Table. 2. Comparative Evaluation

	프로토타이핑 언어	형식 명세	IPOST	본 논문
모델 구축 환경	없음	명세자체가 모델을 표현	OSA 방법론 지원 방법론의 적용 범위가 한정적	S/M의 OOA 방법론 지원 타방법론으로의확장 용이
모델 변환 필요성	변환 필요 수작업에 의한 변환	필요하지 않음 논리 언어 사용명세 자체가 실행	필요하지 않음 도구 자체의 기본 모델 사용	필요하지 않음 분석단계의 결과물인 객체모델이 직접적용
이해 용이성	프로그래밍 언어 습득 필요	수학적인 기호사용 이해의 어려움	형식명세 사용으로 사용자 이해도 저하	다이아그램형태 제공 사용자 관점의 일관성을 위해 명세 도구로 다이아이그램만 사용
시각화	언어에 따라 단순한 사용자 UI만 제공하는 것이 일반적	없음	지원 UI는 개념적으로 제공 객체의 상태 네트웍,관련성	다양한 관점에서 지원 3개의 뷰 제공 시뮬레이션 진행윈도우, 상태머신 윈도우, 행위 윈도우

본 논문에서는 위의 3가지 방법과 본 논문에서 제안한 방법을 비교 평가 하였다. 평가 결과는 <표 2>에서 보는 바와 같이, 개발 과정의 단계의 산물인 객체 모델의 사용으로 개발자와 고객간의 일관된 인터페이스를 유지하므로서 이해도 증진의 장점을 갖는다. 또한 내부 형식 명세 모델을 바탕으로 검증이 가능하고 뿐만 아니라, 이의 시각화로 사용자 확인을 지원한다는 장점을 갖는다.

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 변환기법을 적용하여 상호보완적인 두 방법론의 장점들을 최대화하였다. 이를 위하여 본 논문에서는 객체 모델을 형식 명세 언어인 VDM으로 자동 변환시키기 위한 변환 규칙을 확립하여 변환한 VDM 명세의 객체 지향 모델의 정확성, 일관성, 완전성을 보장한다. 검증이 개발자의 개발 과정에 필요한 것처럼 고객의 요구 사항과의 확인도 개발 초기에 중요한 부분이다. 본 논문에서는 시뮬레이션을 통하여 검증된 객체 모델을 고객에게 확인시키는 방법을 제시하였다. 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 모델의 시뮬레이션을 통한 동적 시각 모형인 프로토타입으로 지원하였다. 시뮬레이션은 객체 모델링의 결과물인 정보 모델과 동적 모델을 참조하여 시스템내의 각 객체간의 상호 작용을 동적인 시각 모형으로 제공하여, 개발자들과 사용자들간의 의사 소통상의 효과를 증진시킬 수 있다.

추후 연구과제로는 형식명세의 정형화와 객관화 방안을 제시할 필요가 있다. 따라서 측정도구에 관한 연구가 요구된다.

참 고 문 헌

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

[2] Latchem, S,. "Component Infrastructures: Placing Software Components in Context," Component_Based Software Engineering, Putting the Pieces Together, Heineman G. and Council, W., ed., Addison-Wesley, 2001.

[3] Atkinson, C., et al., Component-Based Product Line Engineering with UML, Addison Wesley, 2002.

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

University of Queensland.

[5] Rumbaugh, James, Jacobson, Ivar, and Booch, Grady, "The Unified Modeling Language: Reference Manual", 2nd Ed., 2005

[6] Kleppe, Anneke G, et al,. MDA Explained, Addison Wesley, 2003.

[7] Frankel, D, S., Model Driven Architecture-Applying MDA to Enterprise computing, Wiley Publishing, Inc,. 2003.

[8] Braganca, Alexandre and Machado, Ricardo J., "Extending UML 2.0 Metamodel for Complementary Usages of the <<extend>> Relationship within Use Case Variability Specification", SPLC'06, 2006

[9] Metz, Pierre, O'Brien, John and Weber, Wolfgang, "Specifying Use Case Interaction: Types of Alternative Courses", Journal of Object Technology, Vol. 2, No. 2, Mar 2003

저자 소개

임 명 재(종신회원)

• 제 9권 4호 참조
• 현 을지대학교 의료산업학부 교수
<관심분야> SW공학, CBD 방법론, HCI 등

권 영 만(정회원)

• 제 9권 3호 참조
• 현 을지대학교 의료산업학부 교수
<관심분야> 영상처리, 머신비전, 운영체제 등

강 정 진(종신회원)

• 제 9권 4호 참조
• 현 동서울대학 정보통신과 교수
<관심분야> RFID/USN 기술, 디지털무선이동통신 등>