객체지향 동적 모델링 기법의 정형화

김 진 수[†] · 김 정 아[††] · 이 경 환[†††]

요 약

기존에 제안된 객체 모델링 방법론에서 정적 측면의 모델링은 시멘틱 모델 등의 풍부한 시멘틱을 제공하여 모델과 모델링의 많은 부분들을 정형화할 수 있다. 그러나 대부분의 방법론들은 동적 모델과 모델링의 정형화가 미흡하다. 또한 기존의 동적 모델은 실시간과 멀티미디어 시스템에서 매우 중요한 특성인 객체간의 상호작용 관계 및 시간적 제약성을 정확하게 표현할 수 없다.

본 논문에서는 이러한 문제들을 해결하기 위해서 행위를 기반으로한 정형적인 동적 모델과 모델링 절차를 제안한다. 이 모델은 대수구조 개념을 도입하여 객체의 상태 영역을 정의하고, 객체의 행위를 하나의 함수로 정의한다. 또한 이 모델은 시제논리와 정의된 행위함수를 사용하여 객체의 라이프사이클과 병행성을 정형화한다. firing rule들을 사용하여 객체간의 행위적 종속성을 표현하므로써 기존의 객체 중심의 동적 모델에서 표현할 수 없는 시스템 관점의 행위도 일부 표현할 수 있다. 제안된 정형화된 모델을 기반으로 문제를 분석할 수 있는 모델링 도구와 절차를 정형화한다.

Formalization of Object-Oriented Dynamic Modeling Technique

Jin Soo Kim[†] · Jeong Ah Kim[††] · Kyung Whan Lee[†††]

ABSTRACT

In the traditional object modeling methodologies, the object model can be said as formal since it has been based on rich semantic model. But almost of all methodologies lack in formality the dynamic model and modeling process. Dynamic model cannot represent exactly the timing constraints and the interaction among the objects, which are very important features in real-time and multimedia system.

In this paper, we formalize the dynamic model and modeling process based on object behavior and state. This model defines the object state space using the concepts in algebra structure and defines the object behavior function. Also this model can formalize object lifecycle and concurrency among the objects using the temporal logic and behavior function. We apply firing rules to behavior function for modeling the dependency of interaction among the objects.

※본 논문은 관동대 97 학술연구지원비에 의해 연구됨
† 정 회 원:중앙대학교 컴퓨터공학과
†† 정 회 원:관동대학교 컴퓨터교육과
††† 정 회 원:중앙대학교 컴퓨터공학과
논문접수:1997년 2월 5일, 심사완료:1997년 5월 31일

1. 서 론

소프트웨어 개발에 있어서 주어진 문제를 확실하게 이해하고 사용자의 요구사항을 정확하게 추출하는 것은 분석 단계에서 이루어져야 하는 사항들이다. 또한 소프트웨어 개발의 오류가 대부분 개발의 초기 단계에서 발생하고 이러한 오류가 결국은 개발되는 시스템의 신뢰성과 안정성에 커다란 영향을 미치게 된다. 더욱이 분석 단계에서의 오류는 소프트웨어 개발주기의 후반부까지 많은 수정 비용을 초래하게 된다[1]. 따라서 개발자들은 분석 단계에서 오류를 감소시킬 수 있는 많은 방법과 도구에 관심을 갖게 되었다.

특히 실세계를 있는 그대로 모델링 할 수 있는 객체 모델링에 대한 연구가 활발히 진행되어 오면서 분석의 결과를 실세계에 존재하는 정보와 오퍼레이션을 모두 가지고 있는 독립된 요소로서의 객체 단위로 표현한다. 이로써 실세계의 자연스러운 모델링이 된다는 점과, 변화의 가능성이 가장 적은 부분인 객체를 중심으로 분석하므로써 발생될 변화로 인한 파급효과를 최소화 할 수 있다는 장점이 있다[2]. 지금까지 연구된 객체 모델링 결과물들을 분석해보면 공통적으로 구축하는 분석의 기반 개념은 객체 모델이다. 즉, 모든 객체 모델링 방법론들은 객체를 식별해내고, 객체들간의 관련성을 파악하고, 상호 작용을 파악한다. 또한 기존에 제안된 객체 모델링 방법론들은 대부분 시스템을 객체 단위로 모델링하면서 시스템의 다양한 측면을 표현하기 위하여 정적 측면, 동적 측면, 구조적 측면으로 나누어 모델링하므로써 객체에 대한 다양한 모델을 구성할 수 있다는 장점을 갖는다. 특히 정적 측면의 모델링에서는 기존의 구조적 방법에서 사용되던 ER 다이아그램 등의 풍부한 시멘틱을 그대로 사용할 수 있기 때문에 모델과 모델링 방법들의 많은 부분들을 정형화할 수 있다. 그러나 동적 측면의 모델링에서는 다음과 같은 문제점을 갖는다. 첫째, 객체의 동적 특성을 표현하는데 있어서 하나의 객체의 동적 특성은 대부분 상태 전이도로 모델링하고 있다. 기존의 상태 전이도는 시스템 전체를 모델링하는 도구이지만 복잡한 시스템의 경우 상태가 기하 급수적으로 증가한다는 단점을 가지고 있었다. 이를 해결하기 위하여 객체 모델링에서는 하나의 객체에 대한 상태만을 기술한다. 따라서 객체와 객체간의 상호 작용 관계와 시스템 전체의 행위를 모델링하지 못한다는 단점을 유발한다[3]. 둘째, 최근에 실시간 시스템에 대한 개발이 확대되면서, 시스템의 시간적 특성을 기술하는 기법이 요구되고 있다[4]. 기존의 방법론들은 시스템의 시간에 종속된 동적 특성을 상태의 변화 관점으로만 모델링하고 있으므로 객체간의 시간적 제약성을 표현하지 못한다는 단점을 갖는다. 또한 객체간의 병행성이나 동기화도 표현하기 힘들다. 셋째, 대부분의 방법론이 모델링에 중점을 두고 제안되어 모델의 비정형화와 불일치의 문제점과 모델링의 비정형화라는 문제점을 갖고 있다. 이는 여러 평가 논문에서 볼 수 있듯이 동일한 방법론에 대해서 평가하는 사람들마다 서로 다른 평가가 이루어지기도 하고 실제로 방법론이 임의의 개념을 제공하는지의 여부를 판단하지 못하는 경우도 발생한다[5]. 마지막으로, 모든 방법론들이 모델링에 있어 수행할 단계만을 간략하게 제시하고 있으며, 이들 단계의 순차적인 수행을 가정하기 보다는 앞·뒤 단계의 반복적, 순환적 수행을 기본으로 하고 있다. 이는 분석의 각 단계를 수행하는 자세한 방법과 각 단계별 관련성, 전이 방법 등이 명확하게 정의되어 있지 않기 때문이다.

본 논문에서는 이러한 제약점을 해결하기 위하여 행위를 기반으로한 정형화된 동적 모델과 동적 모델링을 제안한다.

2장에서는 제안한 동적 모델의 기반 개념을 살펴보고, 3장에서는 시제논리 및 대수구조 논리를 적용한 객체의 동적 모델을 제안한다. 4장에서는 필요한 모델링 도구와 절차를 정의하고 사례를 보인 다음, 제안한 모델과 모델링을 다른 방법론들과 비교, 평가한다. 마지막으로 5장에서 결론 및 향후연구과제를 기술한다.

2. 관련 연구

본 논문에서는 동적 모델의 정형화를 위하여 이산수학의 기호와 정의, 대수 구조의 정의, 시제논리(temporal logic)의 연산자를 사용하였다. 〈표 1〉은 본 논문에서 사용한 수학 기호와 개념[6, 7]을 나열하였다.

본 논문에서는 대수 구조학에서 정의된 가장 일반적인 개념인 시그네이처(signature)와 동형사상(isomorphism)을 사용한다. 시그네이처란 SIG=(S,OP)로

〈표 1〉 수학 기호의 정의
〈Table 1〉 The definition of mathematical notation

기호	개념
∪	union
∩	intersection
⊆	subset
×	cartesian product
dom r	{x:X | ∃y:Y.<x, y> ∈r, r is relation}
ran r	{y:Y | ∃x:X.<x, y> ∈r, r is relation}
X→Y	Functional relation between X and Y
X→Y	{f:X→Y | finite dom f}

정의하며, 여기서 S는 소트(sort)의 집합을, OP는 상수를 포함한 오퍼레이션 기호들의 집합을 의미한다. A, B를 SIG=(S, OP)를 만족하는 algebra로 가정할 때, 두 algebra 사이에 사상함수(mapping function)가 존재하면 A, B 사이에 준동형사상(homomorphism)이 존재한다고 정의한다. 즉, 모든 s∈S 인 소트에 대해 $f_s: A_s \rightarrow B_s$가 성립하고, 오퍼레이션 집합 OP에 정의된 모든 오퍼레이션 기호에 대해 $f_s(NA)=NB$가 성립할 때 두 algebra 사이에 준동형사상이 존재하는 것으로 정의한다. 사상함수 f_s가 전단사 함수의 특징을 만족할 동형 사상이 존재하는 것으로 정의한다.

또한 시스템의 동적 특성을 분석하고 명세하기 위한 시제 논리에 대한 연구가 활발히 진행되어 왔다[8]. 즉, 시제 논리를 이용하면 동적 행위를 보다 단순하고 정확하게 표현할 수 있다. 시간의 표현에는 시점 기반 표현(point-based representation)과 기간 기반 표현(interval-based representation)으로 나누어 볼 수 있다[9]. 시점 기반 표현에서는 어느 순간에 일어나는 메세지들로 시스템을 표현할 수 있다. 기간 기반 표현의 경우에는 어느 메세지의 발생부터 종료 메세지의 발생까지 기간을 갖는 행위들로의 시스템을 기술하게 된다. 어느 한 관점만으로 객체의 동적 행위를 표현할 수 없다. 왜냐하면 시점 기반 표현의 경우는 시간적 지연을 고려하지 않으므로 객체의 행위 수행의 과정의 분할이나 행위 중첩을 표현할 수 없다. 로봇 시스템의 분석에서도 로봇이 어느 시발점에서 도착지점으로 이동하라는 메세지에 대해, 시발점의 상태에서 도착지점의 상태로 전이하는 과정에서는 엔진의 가동, 경로 추적, 도착 여부의 확인, 엔진 가동 중지 등의 하나의 메세지에 의한 다른 중복되는 메세지를 필요로 하게 되는데, 시점 기반의 경우는 표현할 수 없다. 그러므로 본 논문에서는 시간 표현의 두 개념을 모두 적용한다. 시제 논리의 기술은 일반 상태를 나타내는 기술과 달리 여러 상태의 전이 과정상에서의 특성을 기술할 수 있다는 점에서 객체의 행위를 기술하는데 도움을 줄 수 있다.

본 논문에서 사용하는 시제 논리의 기본 연산자와 그 정의는 〈표 2〉와 같다.

〈표 2〉 시제 논리의 기본 연산자
〈Table 2〉 Basic operators of the temporal logic

오퍼레이터 기호	기술 방법	의미해석
□(always)	□ω	ω는 모든 상태에서 참
○(next)	○ω	다음 상태에서는 ω가 반드시 참
◇(eventually)	◇ω	ω가 어느 상태에서 참을 수 있다.
ρ(precedes)	$\omega_1 \rho \omega_2$	ω_2가 일어나려면, 반드시 ω_1이 ω_2에 선행
μ(until)	$\omega_1 \mu \omega_2$	ω_2가 참인동안은 ω_1도 참

3. 동적 모델의 정형화

객체지향 시스템을 구성하는 객체와 그들간의 관련성을 객체 모델에서 분석, 모델링한 다음 객체에 대한 상태 및 행위 관점의 분석이 필요하다. 이전부터 상태 개념과 더불어 상태 전이 개념은 응용 시스템의 동적 특성을 이해하고 분석하는 중요한 개념으로 간주되어 왔다[10, 11]. 객체도 상태를 가지고 있으며, 현 상태에 따라 객체 모델에 명세한 인터페이스의 반응 시점 및 반응 내용이 결정된다. 그러므로 객체가 시간에 따라 또는 외부와의 상호작용에 따라 변해가는 행위적 특성을 (정의 3-1)의 동적 모델로 분석한다.

(정의 3-1) 동적 모델 $=(D_o, q_0, \Omega_o, B_o, P_o, T_o)$
D_o: 객체의 상태 도메인의 집합
  $D_o = SF \cup USF$
  SF: 만족스러운 상태들의 집합
  USF: 만족스럽지 못한 상태들의 집합
$q_o \in D_o$: 초기상태
Ω_o: 객체의 메세지의 집합

B_o: 객체의 행위의 집합
T_c: 시간 도메인
P_o: $m \in \Omega_o$에 대한 조건식들의 집합으로, m에 대한 상태 조건식이거나 firing rule에 대한 조건식, 관련 시간 제한식 등

객체는 메세지와 소트의 집합체로 분석된 정적구조에 객체의 상태, 입력 메세지, 출력 결과와 시간에 종속적인 속성들로 파악해야 한다. 동적 모델은 현재 상태를 갖는 객체에게 어떤 메세지가 주어졌을 때 이루어지는 객체의 행위와 객체의 다음 상태를 기술하는 것이다.

3.1 상 태

기존의 방법론에서의 상태란 프로그램 경로의 결정 요인이나 값의 범위로 간주되었다. 본 논문에서는 객체의 상태를 수행 가능한 메세지를 처리할 수 있는 시간으로 정의한다. 객체는 어느 시점에서 하나의 상태를 갖는 것으로 전제한다. 객체는 임의의 상태에서 메세지가 도달하기를 기다리므로 상태는 일정 기간에 대응된다. 상태는 다음 처리가능한 메세지를 결정한다. 즉, $\forall m \in \Omega_o . \exists \in D_o$이다. 이를 2장에서 정의한 객체의 정적 모델을 구성하는 Algebra 개념을 기반으로 정의하면 객체의 상태 도메인을 (정의 3-2)와 같이 정의할 수 있다.

(정의 3-2) $\sum = < S_o, \Omega_o >$: 객체에 대한 시그네이처
$\sum_{state} = < S_{state}, \Omega_{state} >$: 상태표현 시그네이처
$\sum_{basic} = < S_{basic}, \Omega_{basic} >$: 기본 소트 시그네이처
i는 식별자이거나 객체 소트 iff, $i \in (S_o - S_{basic})$이면서 $i \leq i'$
s는 상태 소트 iff, $s \in S_{state}, s \leq s'$

S_o={사원, 사원번호, 이름, 나이, 직급, 봉급}인 사원 객체의 예를 들면, $\sum_{basic} = < S_{basic}, \Omega_{basic} >$에서 S_{basic}={이름, 나이, 직급, 봉급}, S_{state}={나이, 직급, 봉급}으로 $S_o - S_{basic}$={사원, 사원번호}이다. 사원과 사원번호는 사원 객체의 식별자이다. Sstate={나이, 직급, 봉급}의 각 소트에 속하는 carrier set으로 객체의 상태가 결정되게 된다.

본 논문에서는 객체의 상태를 표현하기 위하여 논리 연산자에 의해 조합될 수 있는 상태 표현식은 부울(boolean) 값으로 해석될 수 있는 조건식(predicate)으로 간주한다.

(정의 3-3) 객체의 상태 표현식 S(t): predicate, t: 시간 간격

상태 표현식을 조건식으로 정의한 것은 상태가 객체의 상태 전이를 결정하는 요인이며, 도달한 메세지의 수행 여부를 결정짓는 요인이기 때문이다. 상태 표현식을 통해 상태를 저장하고 결정할 객체의 속성을 추가로 식별할 수 있고 이는 객체 모델에 반영되어야 한다. 일반적으로 시스템의 상태란 변수에 의해 표현되어 지는 것으로 상태 변수가 가질 수 있는 값의 영역 즉, 도메인을 갖고 있다. 시스템의 상태는 객체에 의해 가정되어지는 값의 복합체로 정의하고, 이들 객체를 상태 객체로 정의한다. 또한 객체는 속성에 의해 결정된 값의 복합체로 정의하는데, 이들 변수를 상태 변수로 정의한다.

(정의 3-4) t: 상태 표현식 S가 지속될 수 있는 시간 표현식 (lower limit: upper limit) 만약 시간의 제약성을 갖지 않는 경우 lower limit=0, upper limit=∞로 분석한다.

객체의 상태가 변화해가는 과정에서도 항상 만족해야하는 고유 속성을 invariant rule로 정의하고 시제 논리의 "always" 연산자(□)를 이용한다.

(정의 3-5) invariant rule=□ (state expression)
시제 논리 오퍼레이터 □가 적용 될 수 있는 조건식을 통해 객체의 고유 제약사항을 정의한다.

예를들어, 우대 고객 구좌는 반드시 예금액이 2천만원 이상이고, 예치기간이 30일 이상 지나야 한다는 규칙이 있고, 이 규칙이 우대 고객 구좌에 대해서 입금, 출금, 송금 등의 메세지를 처리할 때마다 항상 만족해야 하는 규칙이라면 (정의 3-5)를 이용하여 다음과 같이 나타낼 수 있다.

우대 예금 구좌의 invariant rule=□(balance ≥ 2천

만원 ∧ date ≥ 30일)이다.

3.2 행 위

상태와 미리 정의된 조건에 부합되는 메세지는 해당 객체로 하여금 정해진 행위를 수행하여 자신의 상태를 변화시키거나 상태를 유지하게 한다. 메세지와 연관된 상태 전이를 행위로 정의한다. (정의 3-6)은 행위에 대한 것으로 행위는 원인이 되는 메세지와 행위를 유발하는데 만족되어야 할 조건으로의 firing rule과 일련의 상태 변환 과정과 행위의 결과로 얻어지는 result의 함수로 정의한다.

(정의 3-6) 모든 상태는 정의된 메세지를 받을 수 있으며, 메세지는 처리될 수 있는 firing rule과 firing rule의 만족시 수행될 상태 전이 과정의 행위를 갖는다.

$\forall m \in \Omega_o. \exists s_i, s_j \in D_o.B(s_i, m, f)$

$= s_j \wedge f \in (S_s - D_o)$

S_s : 시스템의 상태 영역

상태 전이에 해당하는 행위 함수를 정의하면

$Behavior(t) : prestate \times m \times f \xrightarrow{\delta} result$

$Behavior(t) \in B_o$

$m \in \Omega_o$: 행위를 유발하는 객체 외부로부터의 메세지

$prestate, result \in D_o$

$f \in (S_s - D_o)$

t : time interval

$\{f\} \cup \{t\} \subseteq P_o$

δ : 상태 전이의 과정

행위를 추상화하여 정의하면

$Behavior(t) : D_o \times P_o \rightarrow D_o$

메세지는 객체의 행위를 발생시킨다. 객체의 상태 전이의 과정은 객체에 정의된 행위에 의해 일련의 상태 변화를 거쳐가는 것이다. 행위는 객체의 활동으로, 하나의 활동은 여러 행동의 조합으로 구성된다. 즉, 행위는 수행에 시간을 요구하지 않는, 시점 기반으로 표현되는 행동들의 합성으로 구성된다.

(정의 3-7) 행위

$행위 = a_1 \circ a_2 \circ ..... \circ a_n$

a_i : 활동

$\exists a_i \in (G \subseteq \Omega_o)$

$\circ$: 합성

상태는 하나의 객체를 중심으로 정의된 메세지 처리에 필요한 객체의 특성을 정의한 것이다. 그러나, 객체의 상태만으로 메세지의 처리가 결정되지는 않는다. 즉, 객체는 하나의 추상화 관점으로 시스템 전체의 상태를 포괄하여 기술할 수 없다. 기존의 구조적 분석 방법론에서의 문제는 시스템 전체의 상태를 파악하기 때문에 상태가 기하급수적으로 증가해 복잡함을 적절히 관리할 수 없다는 점이다. 이를 해결하기 위하여 하나의 객체 단위로 상태 개념을 파악한다. 기존의 객체지향 분석 방법론들[12, 13]에서도 객체 단위로 상태를 파악하도록 정의하고 있다. 그러나 기존 객체지향 분석 방법론의 취약점은 하나의 객체 단위로 상태를 파악하므로써 시스템 전체의 동적 특성에 대한 정보를 얻을 수가 없다는 것이다.

본 논문에서 제안하는 동적 모델에서는 이를 해결하기 위하여 메세지의 처리는 하나의 객체 상태 뿐아니라 메세지에 의한 행위를 유발하는데 필요한 선행 조건을 분석하도록 하며, 이를 메세지와 관련된 firing rule로 간주한다.

(정의 3-8) firing rule : 메세지와 관련된 행위가 수행되는데 필요한 선행 조건으로 다른 객체의 상태 표현식이다.

$\forall O_i \in has\_interaction(O_j). \exists f \in D_{oi}$

D_{oi} : 객체 o_i의 상태 도메인

$has\_interaction(O_j)$: 객체 모델에서 O_j와 관련성을 갖는 객체들의 집합

행위는 메세지 표현식에서 firing rule과 완료 상태를 만족하는 일련의 상태 전이로 상태 표현식과 행위 표현식으로 정의한다. 또한, 실시간 시스템의 경우는 시스템의 결과가 만들어지는데 시간적 제약성을 갖는다. 이는 행위에 시간적 특성을 분석해야 함을 의미한다. 왜냐하면 행위란 결과를 만드는 상태 전이의 연속이기 때문에, 시스템에서 임의의 결과가 어느 정해진 시간에 만들어져야 함은 그 결과를 만들어 내는

에 관련된 행위에 시간적 제약성이 있음을 의미하기 때문이다.

(정의 3-9) Behavior(t), t: 행위의 수행이 완료되어야 하는 시간의 범위를 갖는다.
Behavior(lower limit : upper limit)
만약 시간의 제약성을 갖지 않는 경우
lower limit = 0, upper limit = ∞로 분석한다.

하나의 객체에 대한 일련의 행위는 ○ (next) 연산자를 사용할 수 있는 성질을 갖는다.

(정의 3-10) δ_i, δ_{i+1}: 객체 i에 정의된 행위들로 δ_i 행위 다음에는 수행되어야 할 δ_{i+1}의 행위가 있다.
즉, $\delta_i \rightarrow \bigcirc\ \delta_{i+1}$을 만족하는 행위들을 갖는다.

next 연산자에 의해 연속되는 행위들은 하나의 객체의 라이프사이클을 정의할 수 있다. 상태 전이중에는 하나의 메세지 처리 결과가 또다시 다른 메세지 처리에 의한 상태전이를 유발하는 firing rule에 대응될 수 있다. 즉, 시작 상태에서 완료 상태로의 연속적인 상태 전이를 가질 수 있다. 이를 라이프 사이클로 정의한다.

(정의 3-11) 라이프사이클

$L = \{m_1 m_2 \ldots m_n \mid m_i \in \Omega_o\}$

$L' = \{m_1 m_2 \ldots m_n \mid m_1 \in C,\ C \subseteq \Omega_o,\ m_n \in D,\ D \subseteq \Omega_o,\ m_i \in \Omega_o - (C \cup D),\ 2 \le i \le n-1\}$

$L'' = \{m_1 m_2 \ldots m_n \mid m_1 \in C,\ C \subseteq \Omega_o,\ m_i \in \Omega_o - C,\ 2 \le i \le n\}$

라이프사이클 정의를 바탕으로 행위 함수를 확장할 수 있다.
Behavior(t): $D_o \times P_o \rightarrow D_o$를 Behavior(t): $Do \times P_o^* \rightarrow D_o$로 확장하여 정의할 수 있다.
$s \in D_o$, $p \in P_o$, $r \in P_o^*$로 두고 행위함수(s, λ) = s이며 행위함수(s, rp) = 행위함수(행위함수(s, r), p)이다.
단, λ은 null predicate이다. 또한 다음을 만족해야 한다.
$\forall s \in D_o.\ \exists p \in P_o^*.$(행위함수$(q_0, p) = s$)
단, $q_0 \in D_o$의 초기 상태이며, *는 0번 이상의 반복을 의미한다.

객체의 행위를 메세지, 상태, firing rule, 그리고 결과로 분석한 후 객체들간의 행위의 동기화나 병행성이 검출되어야 한다. 객체의 본질상 독립된 단위로 서로 독립된 제어 흐름을 갖게 된다. 그러므로 정의된 상태와 firing rule이 만족하는 경우, 도달한 메세지에 의해 상태 전이를 언제라도 일으킬 수 있다. 그러나 시스템 전체적인 관점에서 분석하면 서로 독립된 객체가 상호 작용을 일으켜 원하는 결과를 내거나 객체의 상호전이가 어느 정해진 시간내에서 발생되야 하거나 한 객체의 상태 전이 이후에 반드시 다른 객체의 상태 전이가 있어야 시스템의 안정성을 보장하게 된다. 그러므로 상호 독립적인 객체들간의 행위의 상호 연관성을 분석하는 것은 매우 중요한 일이다. 본 논문에서는 객체 상호 작용을 분석하는데 미리 정의된 시제 논리를 사용하므로써 보다 간결하고 정확한 형태의 분석이 되도록 한다.

(정의 3-12) 여러 객체의 행위는 어느 정해진 시간내에 동시에 이루어져야 한다는 성질을 갖기도 한다. 이는 시제 논리의 연산자 μ(until)이 적용되는 성질이다.
δ_i: 객체 i에 정의된 행위, δ_j: 객체 j에 정의된 행위 $\Box(\delta_i\ \mu\ \delta_j)$이면, δ_i와 δ_j는 동시성을 지닌다.

(정의 3-13) 독립된 여러 객체의 행위가 일정한 순서를 갖고 처리되어져야 하는 특성이 존재한다. 이는 시제 논리의 연산자 ρ(precedes)로 분석되는 경우이다.
δ_i: 객체 i에 정의된 행위, δ_j: 객체 j에 정의된 행위 $\Box(\delta_i\ \rho\ \delta_j)$이면, δ_j는 반드시 δ_i 이후에 이루어져야 시스템의 안정성을 보장한다.

행위와 연관된 시간적 요소는 시제 논리를 사용할 수도 있다. 이는 연산자 ◇(eventually)를 이용하여 분석한다. 즉, 시간과 관련지워진 행위 수행후 일정 시간이 지나면 결과적으로 (◇) 또다른 행위로의 전이가 이루어짐을 분석하면 된다.

(정의 3-14) 행위와 연관된 시간적 제약성

δ, ρ : 행위
t : time point
T : absolute time
$(\delta \wedge t=T) \rightarrow \Diamond(\rho \wedge t=T+time\_limit)$

(정의 3-15) 시제 연산자는 조합하여 분석에 사용될 수 있다.

이번 장에서는 상태와 행위를 중심으로 객체의 동적 특성을 분석하기 위한 동적 모델과 필요한 기호들을 정의하였다. 다음 장에서는 정의된 기호와 모델을 바탕으로 동적 모델링 과정을 설명하고 적용 예를 보인다.

4. 행위를 기반으로한 동적 모델링

4.1 동적 모델링 과정

정적 모델링을 통하여 객체의 구조와 객체간의 관련성을 모델링하면 이를 바탕으로 동적 모델링에서는 객체의 시간에 따른 동적 특성을 분석하여 모델링한다. 먼저 (정의 3-2)의 모델에 근거하여 정적 모델에서 분석된 객체와 관련된 상태 도메인을 〈표 3〉의 상태 기술서에 작성한다. 이 상태 기술서에 기술된 상태 도메인은 객체에 정의된 적어도 하나 이상의 메세지 처리와 관련 지어질 수 있는 것이어야 하며, 메세지 관점에서는 하나 이상의 상태 도메인과 관련될 수 있다. 이 과정은 (그림 1)과 같이 이루어진다.

〈표 3〉 각 객체별 상태 기술서
〈Table 3〉 State script for each object

객체이름	상태도메인

메세지와 관련된 행위를 기술하기 위하여 (정의 3-1)의 동적 모델과 (정의 3-6)의 행위 정의에 따라서 〈표 4〉의 행위 기술서를 작성한다.

〈표 4〉 행위 기술서
〈Table 4〉 Behavior script

객체 이름	메세지 이름	초기 상태 도메인	firing rule	처리후 상태

상태 기술서와 행위 기술서를 이용하여 OSTD를 작성하는 규칙은 다음과 같이 정의한다.

〈규칙 1〉 객체의 모든 상태들은 초기 상태(q_0)로 부터의 메세지와 firing rule의 집합으로 구성된

```
입력 : So, Ωo
출력 : Do,Po,To

1. for each s ∈ So.
   if( s has value boundary ) then make each boundary as state
   else if (동작의 결과를 명시할 경우 또는 동작 중인 상태)
        then    So  = So U ( status )
                Do  = Do U ( status 와 관련된 상태 )
2. for each d ∈ Do
    if(관련된 메세지가 존재)
    then { 행위 기술서에 추가
           타임테이블에 해당 상태의 지속 시간을 기술 }
    else 상태 도메인에서 제거
3. for each m ∈ Ωo , 1 ≤ i ≤ n
        3.1  m의 처리에 필요한 자신의 상태(s)를 정의
        3.2  if(~(s∈Do)) then Do = Do U {s}
        3.3  m의 처리에 필요한 환경적 상태(f)의 정의
        3.4  if(f ∈ 다른 객체의 상태) then 행위 기술서에 추가
```

(그림 1) 상태기술서로부터 동적 상태 파악의 단계
(Fig. 1) Procedure of the dynamic state analysis from the state script

경로를 가지며 이는 선형 구조를 갖는다.

〈규칙 2〉 반복하여 수행되는 행위와 일련의 행위들은 사이클로 나타낼 수 있다.

〈규칙 3〉 원하지 않은 객체의 상태로부터 바람직한 객체의 상태로의 경로를 반드시 정의한다.

〈규칙 4〉 다른 객체와 병행 수행되는 경우는 shared transition으로 간주하여, $\delta_i (1 \le i \le n)$를 메세지 이름 앞에 접속하고, 타이밍 다이아그램에 정보를 기록한다.

이 규칙에 따라 문제를 분석하므로써 (정의 3-2)와 (정의 3-6)에 의한 상태와 행위 및 행위 처리에 필요한 firing rule을 추출하게 된다. 행위 기술서를 작성하는 과정을 정형화하여 기술하면 (그림 2)와 같다.

```
입력 : D0, Ω0, P0, T0
출력 : OSTD (객체들의 상태 전이도)

Mt : 객체 t 에 정의된 메세지 집합
q0 : 초기 상태
USF : 만족스럽지 못한 상태들의 집합
SF : 만족스러운 상태들의 집합
1. OSTD0 = ∅
2. OSTDi : 이미 구성된 상태와 행위 집합, i ≥ 0
   2.1 다음을 만족하는 집합 T를 구한다.
       T = { s' : for some s∈OSTDi and q0,
             behavior(s,f,m) = s',s' ∉ OSTDi }
       OSTDi+1 = OSTDi ∪ {T}
   2.2 다음을 만족하는 집합 T를 구한다.
       T = { s' : for some s∈OSTDi and q0,
             behavior(s,f,m) = s', s' ∈ OSTDi}
       OSTDi+1 = OSTDi ∪ {T}  // Cyclic Property
   2.3 for each s ∈ USF, 다음을 구한다.
       T = { s' : for some s∈USF and q0 ,
             behavior(s,f,m) = s', s' ∈ SF}
       OSTDi+1 = OSTDi ∪ {T}
   2.4 if(f != NULL)
       then { firing rule에 명시된 상태와 행위간의
              시간적 관련성을 타이밍 다이아그램에 작성 }
3. if OSTDi = OSTDi+1
   then   OSTD = OSTDi
          stop
   else   goto 2
```

(그림 2) 행위 기술서를 작성하는 과정
(Fig. 2) Procedure of the constructing behavior script

행위 기술서에 담긴 객체의 동적 특성을 시각적으로 표현하기 위하여 기존의 상태 전이 다이아그램을 변형하여 객체 상태 전이 다이아그램(OSTD : Object State Transition Diagram)을 제안한다.

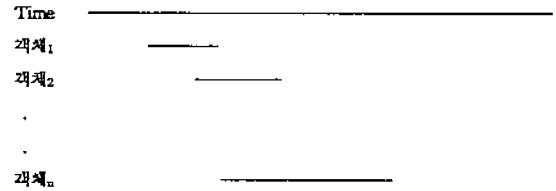
동적 모델을 구성하는 시간적 요소에 대한 모델링 도구는 타임 테이블과 타이밍 다이아그램으로 나눌 수 있다. 타임 테이블은 객체의 상태와 행위에 연관된 절대 시간을 표현하기 위한 모델링 도구이며, 객체의 상태나 행위에 분석된 시간적 제약성을 표현한다. 〈표 5〉는 타임 테이블의 기본 형태이다.

〈표 5〉 타임 테이블의 기본 형태
〈Table 5〉 Basic form of time table

객체 이름	상태 도메인/행위	Lower Limit	Upper Limit

타이밍 다이아그램은 서로 다른 객체들간의 상호 작용을 표현하기 위한 행위의 동기화 또는 순서화의 특성을 표현하기 위한 모델링 도구이다. 객체별 시간적 모델링과 별도로 객체들 간의 상호 관련성을 모델링한다.

타이밍 다이아그램은 x축에 상태적 시간을 표현하며 y축에 상호 시간적 관련성을 갖는 객체로 구성한다. 이는 Booch가 제안한 시간적 객체 행위의 순서화만을 표현할 수 있는 하드웨어 타이밍 다이아그램보다 시간에 관련된 많은 정보를 표현 할 수 있다는 장점을 갖는다. 타이밍 다이아그램의 기본 형태는 (그림 3)과 같다.

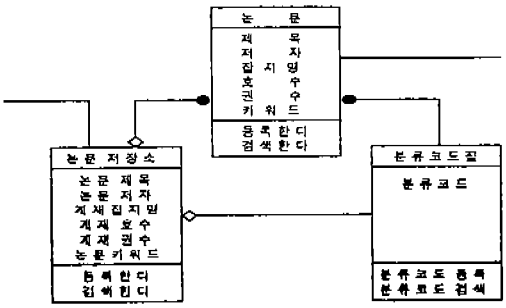


(그림 3) 타이밍 다이아그램
(Fig. 3) Timing diagram

4.2 적용 예

새로운 논문을 원하는 위치에 두었다가 원할 때 해당하는 논문을 검색할 수 있게 해주는 논문검색 시스템에 대한 문제에 대해서 정형화된 동적 모델을 이용하여 행위를 기반으로한 동적 모델링하는 과정을 보여준다. 먼저 주어진 문제에 대한 정적 모델링을 수

행하여 객체를 추출해야 한다. 다음 (그림 4)는 정적 모델링에서 추출된 객체 모델을 Rumbaugh의 OMT 표기법으로 표현하였다.



(그림 4) 논문 검색시스템의 부분적인 객체 모델
(Fig. 4) Partial object model for paper retrieval system

(그림 4)의 객체 모델에서 추출된 논문 저장소 객체와 논문 객체의 상태 기술서는 다음 〈표 6〉과 같다.

〈표 6〉 추출된 객체들의 상태 기술서
〈Table 6〉 State script for derived objects

객체이름	상태 도메인
논문 저장소	count = 0 ∨ 0 < count < max − 1 ∨ count = max ∨ count > max
논문	new, classified

논문 저장소 객체에는 네가지 상태가 있고 논문 객체에는 두가지 상태가 있다. 각 객체에 대한 행위 기술서들은 각각 다음 〈표 7〉, 〈표 8〉, 〈표 9〉와 같다.

〈표 7〉 추출된 논문 객체의 행위 기술서
〈Table 7〉 Behavior script for derived paper object

객체 이름	메세지이름	초기상태도메인	Firing Rule	처리후상태
논문	논문등록	New		Classified
	논문검색	Classified		Classified

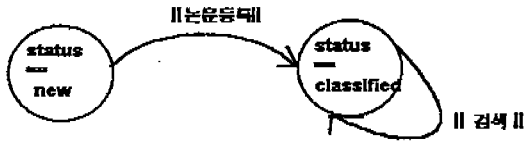
〈표 8〉 추출된 논문저장소 객체의 행위 기술서
〈Table 8〉 Behavior script for derived paper_repository object

객체 이름	메세지이름	초기상태도메인	Firing Rule	처리후상태
논문저장소	논문등록	Empty ∨ !Empty	논문 == Classified	!Empty ∨ full
	논문검색	!Empty	논문 == Classified	!Empty

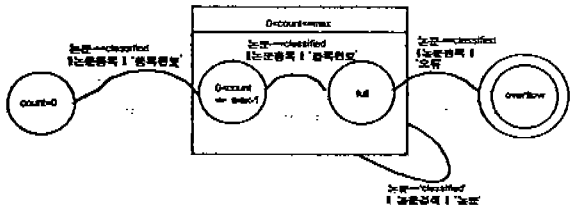
〈표 9〉 추출된 분류코드집 객체의 행위 기술서
〈〈Table 9〉 Behavior script for derived classified_code_file objects

객체 이름	메세지이름	초기상태도메인	Firing Rule	처리후상태
분류코드집	코드등록	Empty ∨ !Empty	코드 == inserted	!Empty ∨ full
	코드검색	!Empty	코드 == inserted	!Empty

작성된 행위 기술서를 기반으로 논문 객체에 대한 OSTD는 다음 (그림 5)와 같고 논문 저장소 객체에 대한 OSTD는 (그림 6)과 같다.



(그림 5) 논문 객체에 대한 OSTD
(Fig. 5) OSTD for paper object



(그림 6) 논문 저장소 객체에 대한 OSTD
(Fig. 6) OSTD for paper_repository object

(그림 6)의 경우, (그림 3)에 따라 0 < count < = max의 상태를 abstract 상태로 정의하고, 0 < count < = max-1과 full의 두 상태로 분할하였다. 이 과정에서 0 < count < = max-1의 상태에서 '논문등록' 메세지에 의해 full로 전이하기 위해서는 시스템내의 다른 클래스인 논문이 'classified'된 상태여야 함을 firing으로 정의하고 있다. 이로써 논문의 상태와 논문 저장소의 상태 확인을 통한 두 객체간의 상호작용 관계를 표현할 수 있다. 즉, 서로 다른 두 객체간의 상태 종속성을 firing rule로 표현하므로써 시스템의 전체적인 행위를 모델링할 수 있게 된다.

4.3 평 가

본 논문에서 제안한 행위 기반의 정형화된 동적 모델과 모델링 방법을 객관적으로 평가하기 위하여 기존의 평가 방법을 적용하여 평가한다. 기존에 제안된 객체지향 방법론들의 평가를 위한 여러 기준들이 제시되었다. Champeaux에 의한 방법[5], Fichman에 의한 방법[14], Monarchi에 의해 제안된 비교 항목[15]에 의한 방법들이 있다. 각 평가 방법들의 공통점은 객체지향 분석 또는 설계 방법론들이 제공하는 용어 수준의 개념과 다이아그램들을 비교의 항목으로 정하고, 각 방법론들이 어느 정도를 제공하는가를 평가하고 있다. 이는 각 방법들을 동일한 조건 상에서 동일한 프로젝트를 동시에 수행하면서 비교하기에는 방법론들이 너무 많은 비교 요소들을 갖고 있기 때문에 실질적인 평가가 어렵다[14]. 그러므로 개념, 기호들, 이해도, 방법론의 지침들을 비교하므로써 개발될 소프트웨어의 생산성, 품질, 수정 용이성 등의 특성을 얼마만큼 향상시킬 수 있는지를 파악하는 것이 바람직하다. 본 논문에서는 Monarchi에 의한 방법에 의해서 동적 모델링 부분에 한하여 평가를 하였다.

Monarchi는 기존의 방법들을 분석과 설계 지원 방법으로 구분하고, 각 단계별로 평가 항목을 선정하였다. 다른 방법에 비해 본 논문에서 제안하는 행위를 기반으로한 동적 모델링이 시간과 관련하여 상태에 연관된 시간적 제약성과 전이에 연관된 제약성 모두를 파악하도록 하고 있고, 다른 방법들은 객체의 분할을 통한 시스템의 복잡도 관리만을 제공하지만 하나의 객체내에서도 동적 구조의 분할을 가능하게 하므로써 한 객체의 복잡도도 관리할 수 있도록 하고 있다.

기존의 방법론중 Rumbaugh[12]의 OMT의 경우, 정적 모델의 객체와 동적 모델간의 명확한 관련을 규정할 수 없다. 특히 Shlaer/Mellor[13], Coad/Yourdon [16]은 동기화나 행위 분할 등을 지원하지 못하는 단점이 있다. Rumbaugh의 OMT와 James Martin[17]의 OOIE는 행위 분할은 지원되나 동기화나 병행성에 대한 개념이 약하다. Booch[18]는 동기화와 병행성은 지원되나 행위 분할의 개념이 지원되지 않는다.

또한 동적인 분석을 통하여 하나의 클래스를 동적 특성을 달리하는 서로 다른 클래스로 분리하는 방법을 통해 보다 명확하게 시스템을 이루는 객체의 특성을 파악할 수 있게 하고, 시제 논리의 사용을 통해 동적 모델의 정형화를 기할 수 있다는 장점을 지닌다. 또한 firing rule 개념의 사용으로 다른 객체와의 상호작용을 명백하게 분석할 수 있도록 하였다.

본 논문에서는 행위를 기반으로한 새로운 클래스의 추출 방법을 제안하였다. 이는 하나의 객체에 정의된 상태 도메인에 따라 동일한 오퍼레이션에 대해 객체의 행위가 명백하게 서로 다르게 분석되었다면 이를 서로 다른 클래스로 식별하므로써 문제가 지닌 동적인 특성을 OSTD의 참조 없이도 이해할 수 있도록 하고 있다. 또한 기존의 방법론들은 동적인 특성에 의한 모델링을 단지 상태 전이도를 통한 모델링으로 국한 하고 있으나 본 논문에서는 OSTD를 통한 모델링의 결과로 동적 특성을 반영하는 클래스의 식별을 유도하므로써 문제의 동적 특성을 보다 명확하게 하는 장점을 갖는다.

5. 결 론

기존에 제안된 객체 모델링 방법론들은 대부분 시스템을 객체 단위로 모델링하면서 시스템의 다양한 측면을 표현하기 위하여 정적 측면, 동적 측면, 구조적 측면으로 나누어 모델링하므로써 객체에 대한 다양한 모델을 구성할 수 있다는 장점을 갖는다. 특히 정적 측면의 모델링에서는 기존의 구조적 방법에서 사용되던 ER 다이아그램 등의 풍부한 시멘틱을 그대로 사용할 수 있기 때문에 모델과 모델링 방법들의 많은 부분들을 정형화할 수 있다. 그러나 동적 측면의 모델링에서는 객체와 객체간의 상호 작용 관계와 시스템 전체의 행위를 모델링하지 못하고, 객체간의 시간적 제약성을 표현하지 못하며 객체간의 병행성이나 동기화도 표현하기 힘들다는 문제점을 가지고 있었다. 또한 대부분의 방법론이 모델링에 중점을 두고 제안되어 모델간의 비정형화와 불일치의 문제점과 모델링의 비정형화라는 문제점도 가지고 있다.

본 논문에서는 이러한 제약점을 다음과 같이 정형화하여 해결하였다. 첫째, 대수구조에 있는 basic sort의 개념과 state sort라는 개념을 도입하여 객체의 상태 영역을 정의하였고, 둘째, 객체가 가질 수 있는 invariant rule을 시제논리를 이용하여 표현하였다. 셋째, 객체의 한 상태에서 메세지를 받아서 다음 상태

까지의 interval을 행위라는 함수로 정의하고 행위함수를 기반으로 관련성을 추출하도록 지원하며, 넷째, 시제논리를 이용하여 객체들간의 동기화와 객체의 라이프사이클을 정형화하였다. 다섯째, 한 객체와 다른 객체간의 관련성을 firing rule로 표현하였다.

앞으로 이를 바탕으로 동적 모델의 검증 방법과 동적 모델 구축 환경 및 동적 모델과 정적 모델간의 연계성에 관한 연구가 보강되어야 한다.

참 고 문 헌

[1] Robert H. Bourdeau and Betty H. Cheng, "A Formal Semantic for Object Model Diagrams," IEEE Trans. on SE, Vol. 21, No. 10, pp. 799-821, 1995.

[2] 이경환, 객체모델링 방법론, 중앙대학교 출판국, 1996.

[3] Martin Fouwler, "A Comparison of Object-Oriented Analysis and Design Method," OOPSLA'92, Tutorial, July, 1992.

[4] Sidney C. Bailin, "An Object-Oriented Requirements Specification Method," Communication of ACM, Vol. 32, No. 5, pp. 608-623, 1989.

[5] D.De Champeaux and P. Faure, "A Comparative Study of Object-Oriented Analysis Methods," JOOP, Vol. 5, No. 1, pp. 21-33, 1992.

[6] Tim Denvir, Introduction to Discrete Mathematics for Software Engineering, Macmillan Computer Science Series, 1986.

[7] Bertrand Meyer, Introduction to the Theory of Programming Languages, Englewood Cliffs, NJ, Prentice Hall, 1990.

[8] B. C. Moszkowski, Executing Temporal Logic Programs, Cambridge University Press, 1986.

[9] Shem-Tow Levi and Ahok K. AgraWaka, Real-Time system Design, McGraw-Hill Publishing Company, 1990.

[10] J. C. Huang, "State Constraints and Pathwise Decomposition of Program," IEEE Trans. on SE, Vol. 16, No. 8, pp. 880-896, 1990.

[11] Stanley Lee and Suzanne Sluizer, "An Executable Language for Modeling Simple Behavior," IEEE Trans. on SE, Vol. 17, No. 6, pp. 527-543, 1991.

[12] James Rumbaugh, et. al, Object-Oriented Modeling and Design, Prentice Hall, 1990.

[13] Sally Shlaer and Stephen J. Mellor, Object Lifecycles : Modeling the World in States, Yourdon Press, Englewood Cliffs, NJ, Prentice Hall, 1992.

[14] R. G. Fichman and C. F. Kemerer, "Object-Oriented and Conventional Analysis and Development Methodologies : Comparison and Critique," IEEE Computer, Oct., pp. 22-39, 1992.

[15] D. E. Monarchi and G. I. Puhr, "A Research Typology for Object-Oriented Analysis and Design," Communication of ACM, Vol. 35, No. 9, pp. 35-47, 1992.

[16] Peter Coad and Edward Yourdon, Object-Oriented Analysis, Prentice-Hall Inc., 1990.

[17] James Martin and James J. Odell, Object-Oriented Analysis and Design, Prentice-Hall Inc., 1992.

[18] Grady Booch, Object-Oriented Analysis and Design with Application, 2nd Edition, The Benjamin/Cummings Publishing Company, Inc., 1994.

김 진 수

1986년 중앙대학교 공과대학 전자계산학과 졸업(이학사)

1988년 중앙대학교 대학원 전자계산학과 졸업(이학석사)

1995년~1997년 중앙대학교 기술과학연구소 연구원

현재 중앙대학교 대학원 컴퓨터공학과 졸업예정(공학박사)

관심분야 : 소프트웨어 공학, 객체지향 방법론, 정형화 기법, 하이퍼미디어 시스템

김 정 아

1988년 중앙대학교 전산과 졸업(이학사)
1990년 중앙대학교 대학원 전자계산학과 졸업(공학석사)
1994년 중앙대학교 대학원 전자계산학과 졸업(공학박사)
1994년~1996년 2월 중앙대학교 Post-Doc.
1996년 2월~현재 관동대학교 컴퓨터 교육과 조교수
관심분야:소프트웨어 공학, 재사용 이론, 형식 명세 기법, 객체지향 개발 방법론

이 경 환

1980년 중앙대학교 대학원 응용수학 전공(이학박사)
1982년~1983년 미국 Aurban대학 객원교수
1986년 서독 Bonn대학 객원 교수
1971년~현재 중앙대학교 컴퓨터공학과 교수
관심분야:소프트웨어 공학, 객체지향 모델링, 소프트웨어 재사용.