UML 모델의 저장 및 질의를 위한 관계형 데이터베이스 설계 및 구현

이 성 대　　박 휴 찬
한국해양대학교 컴퓨터공학과
e-mail: daeya@ce.kmaritime.ac.kr, hcpark@hanara.kmaritime.ac.kr

Design and Implementation of the Relational Database for Storing and Querying UML Models

Seong-Dae Lee, Hyu-Chan Park
Dept. of Computer Engineering, Korea Maritime University

요 약

UML(Unified Modeling Language)은 OMG(Object Management Group)에서 표준으로 지정한 통합된 시스템 개발방법론이다. 특히, 소프트웨어 시스템의 설계 및 개발 등을 체계적으로 지원하는 모델링 언어이다. 이러한 UML로 개발된 모델들의 효율적인 관리를 위하여 통합하여 저장하고 관리하는 것이 필요하다. 이를 위하여 본 논문에서는 UML을 관계형 데이터베이스로 사상시키고 질의하는 알고리즘을 제안한다. 제안한 알고리즘은 UML 모델들을 다수의 사용자가 서로 공유하도록 하여 시스템 개발 분야에서 모델의 재사용과 모델정보의 검색을 보다 효율적으로 수행할 수 있도록 한다.

1. 서론

1989년 설립된 OMG(Object Management Group)의 목표는 분산객체어플리케이션 개발을 위한 공통 프레임워크 가이드라인과 상세한 객체관리 규약을 정립하는 것이다[1]. OMG에서 추구하는 것 중의 하나가 본 논문에서 다루는 UML(Unified Modeling Language)이다. UML은 Rumbaugh의 OMT 방법론, Booch의 Booch 방법론, 그리고 Jacobson의 OOSE 방법론 등을 통합하여 만든 통합 모델링 방법이다. 시스템 분석 및 설계 방법론의 표준 지정을 목표로 제안되었고, 1997년 11월 UML version 1.1을 시작으로 1999년 8월 UML version 2.0에 이르게 되었으며, 이미 많은 시스템 개발 도구들이 UML을 지원하고 있다[4].

특히, 소프트웨어 개발 분야에서는 개발할 시스템의 구성요소를 체계적으로 분석하고 설계하여, 이를 효율적으로 저장하고, 운용 필요에 따라서는 통합하여 관리하는 것이 요구된다.

본 논문에서는 UML을 사용하여 설계한 모델을 다수의 사용자가 공유하고 신속한 검색을 위하여 관계형 데이터베이스에 저장하고 검색하는 알고리즘을 제안한다. 즉, UML의 구성요소인 클래스 다이어그램, 클래스, 애트리뷰트, 오퍼레이션, 관계를 각각 집합으로 사상한다. 다시, 이러한 집합들을 관계형 데이터베이스의 테이블들로 변환한다. 각각의 테이블이 지니고 있는 상호 연관성에 의해서 사용자가 원하는 클래스 다이어그램 정보를 검색할 수 있게 된다. 이 상호 연관성은 관계형 테이블의 기본 키(primary key)와 참조 키(foreign key)가 된다[3].

본 논문의 구성은 2장에서 UML의 구성요소와 클래스 다이어그램에 대해 간략하게 설명하고, 3장에서는 제안하는 관계형 데이터베이스로의 사상 및 질의 알고리즘에 대해서 설명한다. 4장에서는 결론과 함께 추후 연구과제에 대해서 논의한다.

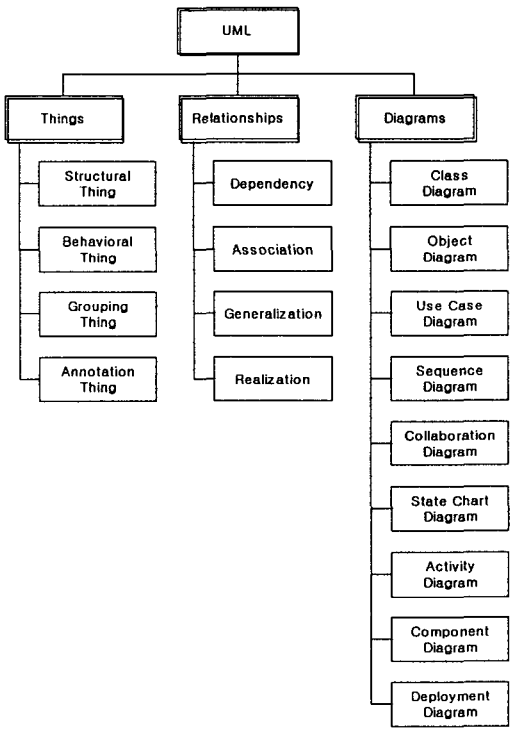
2. UML

UML은 통합된 시스템 개발방법론으로서 시스템을 가시화·명세화·문서화하는 것이다. 특히 기존에 존재하던 여러 개발방법론들의 장점들을 모두 수용했으며, 확장이 용이하고, 다양한 표기법을 가지고 있어 실시간 시스템뿐 아니라 여러 종류의 시스템 개발에 사용되어 질 수 있다.

본 장에서는 UML의 구성요소에 대하여 먼저 살펴보고, 다음으로 본 논문에서 다루게될 클래스 다이어그램(Class Diagram)에 대해 설명한다.

2.1 UML의 구성 요소

UML은 (그림 1)과 같이 크게 3가지의 구성요소로 구분할 수 있다.



(그림 1) UML 구성 요소

첫 번째 구성요소는 사물(Things)로서 Structural Thing, Behavioral Thing, 그리고 Annotation Thing으로 나뉜다. 두 번째 구성요소는 관계(Relationships)로서 의존(Dependency) 관계, 연관(Association) 관계, 일반화(Generalization) 관계 그리고 실제화(Realization) 관계로 다시 나눌 수 있다. 마지막 구성요소로는 설계를 시각화하는 다이어그램이다. 다이어그램을 다시 세분화하면 정적구조(Static Structure) 다이어그램, 사용 사례(Use Case) 다이어그램, 순차(Sequence) 다이어그램, 협력(Collaboration) 다이어그램, 상태도표(State Chart) 다이어그램, 활동(Activity) 다이어그램, 구현(Implementation) 다이어그램 등이 있다. 다시, 정적구조(Static Structure) 다이어그램은 클래스(Class) 다이어그램, 객체(Object) 다이어그램으로, 구현(Implementation) 다이어그램은 컴포넌트(Component) 다이어그램, 배치(Deployment) 다이어그램으로 나눌 수 있다.

2.2 클래스 다이어그램(Class Diagram)

클래스 다이어그램은 정적인 구조 모델을 표현하는 다이어그램으로 클래스, 인터페이스(Interface)와 같은 모델 요소와 모델 요소간의 관계들의 집합으로 정의할 수 있다[1].

클래스는 사물을 추상화하는 것으로 이름(Name)과 애트리뷰트 리스트(Attribute list) 및 오퍼레이션 리스트(Operation list)로 구성된다. 관계는 클래스와 클래스, 클래스와 객체간의 상호 연관성을 나타낸다.

본 논문에서는 UML의 많은 다이어그램들 중에서 클래스 다이어그램을 관계형 데이터베이스에 저장하고 검색하기 위한 알고리즘을 제안한다.

3. UML 모델의 저장 및 질의 알고리즘

UML 클래스 다이어그램은 계층적인 구조를 지니고 있다. 즉, 하나의 클래스 다이어그램은 다수의 클래스들로 구성되며, 각 클래스 내부에는 다수의 애트리뷰트(Attibute)와 오퍼레이션(Operation)이 존재한다.

관계형 데이터베이스에서는 이러한 계층적인 구조를 허용하지 않으므로 클래스 다이어그램을 관계형 스키마로 변환하는 규칙들을 정의한다. 또한, 이러한 스키마에 따라 저장된 데이터베이스에서 클래스 다이어그램을 질의하는 방법에 대해서 논의한다.

3.1 사상 규칙

개발하고자 하는 전체 시스템에 대한 설계는 여러 개의 클래스 다이어그램으로 모델링된다. 모델링된 클래스 다이어그램들을 관계형 스키마에 사상시키기 위한 규칙들을 다음과 같이 정의한다.

[규칙 1] *클래스 다이어그램 집합* $D = \{d_1, d_2, ..., d_n\}$으로 정의하고, D의 임의의 원소 d_x는 클래스 다이어그램의 식별자 또는 이름이다.

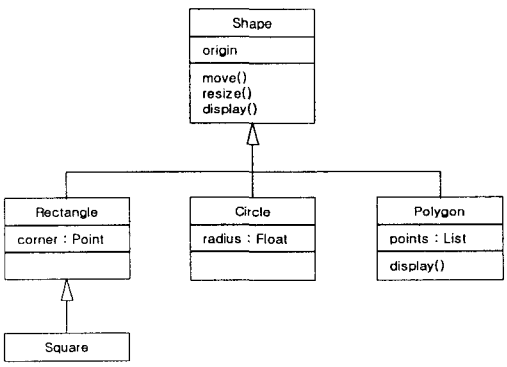
[규칙 2] 클래스 다이어그램 내의 *클래스 집합* $C = \{c_1, c_2, ..., c_n\}$으로 정의하고, C의 임의의 원소인 클래스 $c_x = (class_name, d_i)$으로 정의한다. 여기서, d_i는 c_x가 소속된 상위 D의 원소이다.

[규칙 3] 클래스 내부의 *애트리뷰트 집합* $A = \{a_1, a_2, ..., a_n\}$으로 정의하고, A의 임의의 원소인 애트리뷰트 $a_x = (attribute_name, c_i)$으로 정의한다. 여기서, c_i는 a_x가 소속된 상위 C의 원소이다.

[규칙 4] 클래스 내부의 *오퍼레이션 집합* $O = \{o_1, o_2, ..., o_n\}$으로 정의하고, O의 임의의 원소인 오퍼레이션 $o_x = (operation_name, c_i)$으로 정의한다.

각 클래스의 상속성과 클래스간의 연관성를 설정하는 것을 관계라고 한다면, 관계 집합은 다음과 같이 정의할 수 있다.

[규칙 5] 클래스 다이어그램 내부의 클래스간의 관계 집합 $R = \{r_1, r_2, ..., r_n\}$으로 정의하고, R의 임의의 원소인 관계 $r_x = (c_i, c_j, relationship)$으로 정의한다. 즉, 클래스 c_i와 클래스 c_j는 *relationship*의 관계가 있음을 나타낸다. 여기서, c_i는 상위 클래스이고 c_j는 하위 클래스이다.



(그림 2) UML 클래스 다이어그램 예제 (Graphic)

(그림 2)는 'Graphic'과 관련된 도형들을 UML로 모델링한 클래스 다이어그램이다. 이 그림에서 삼각형 화살표는 일반화(Generalization)를 의미한다. 일반화는 클래스에서 상속(Inheritance)과 동일한 의미를 가진다. 가장 상위의 클래스 'Shape'를 'Rectangle', 'Circle', 'Polygon' 클래스들이 상속한다는 의미이며, 'Square' 클래스는 'Rectangle'에서 상속받는다.

(예제 1) (그림 2)의 클래스 다이어그램에 본 논문에서 정의한 규칙을 적용하면 다음과 같은 집합들을 구할 수 있다. 클래스 다이어그램 집합 D = {*Graphic*}, 클래스 집합 C = {(*Shape, Graphic*), (*Rectangle, Graphic*), (*Circle, Graphic*), (*Polygon, Graphic*), (*Square, Graphic*)}, 애트리뷰트 집합 A = {(*origin, Shape*), (*corner, Rectangle*), (*radius, Circle*), (*points, Polygon*)}, 오퍼레이션 집합 O = {(*move, Shape*), (*resize, Shape*), (*display, Shape*), (*display, Polygon*)}, 관계 집합 R = {(*Shape, Rectangle, generalization*), (*Shape, Circle, generalization*), (*Shape, Polygon, generalization*), (*Rectangle, Square, generalization*)}으로 나타낼 수 있다.

본 논문에서는 각 집합내의 모호성(ambiguity)을 제거하기 위하여 규칙들을 집합의 원소 수준까지 확장하였다.

3.2 관계형 데이터베이스 생성 규칙

사상된 집합에 대하여 다음의 규칙을 적용하면 데이터베이스로 변환할 수 있다.

[규칙 6] 각 집합 D, C, A, O, R은 관계형 모델(relational model)에서 각각 하나의 관계형 스키마(relational schema)로 변환한다.

규칙 6을 적용하면 클래스 다이어그램은 다음의 다섯 개의 관계형 스키마로 변환된다.

D-schema = (*diagram_name*)
C-schema = (*class_name, diagram_name*)
A-schema = (*attribute_name, class_name*)
O-schema = (*operation_name, class_name*)
R-schema = (*super_class, sub_class, relationship*)

[규칙 7] 각 집합의 원소는 [규칙 6]을 통하여 생성된 관계형 스키마의 튜플(tuple)로 변환된다.

다음의 (그림 3)은 위의 관계형 스키마를 사용하여 (그림 2)의 예제를 테이블로 변환한 것이다.

D (Diagram) Table

diagram_name
Graphic

C (Class) Table

class_name	*diagram_name*
Shape	Graphic
Rectangle	Graphic
Circle	Graphic
Polygon	Graphic
Square	Graphic

A (Attribute) Table

attribute_name	*class_name*
origin	Shape
corner	Rectangle
radius	Circle
points	Polygon

O (Operation) Table

operation_name	*class_name*
move	Shape
resize	Shape
display	Shape
display	polygon

R (Relatoinship) Table

super_class	*sub_class*	*relationship*
Shape	Rectangle	generation
Shape	Circle	generation
Shape	Polygon	generation
Rectangle	Square	generation

(그림3) 변환된 데이터베이스 테이블

3.3 UML 클래스 다이어그램에 대한 질의

본 절에서는 사상규칙에 의해서 변환된 데이터베이스에서 클래스 다이어그램을 검색하는 질의 방법을 설명한다.

<표 1> 클래스 다이어그램을 검색하는 질의

①	클래스 다이어그램에 속한 모든 클래스 이름을 검색한다. SELECT *C.class_name* FROM *C, D* WHERE $C.d_i$ = $D.d_x$
②	각 클래스 내부의 애트리뷰트들을 검색한다. SELECT *A.attribute_name* FROM *A, C* WHERE $A.c_i$ = *C.class_name* 각 클래스 내부의 오퍼레이션들을 검색한다. SELECT *O.operation_name* FROM *O, C* WHERE $O.c_i$ = *C.class_name*
③	클래스간의 관계들을 검색한다. SELECT *R.relationship* FROM *R, C* WHERE (*R.*c_i = *C.class_name1* AND *R.*c_i = *C.class_name2)* OR (*R.*c_j = *C.class_name1 AND R.*c_i = *C.class_name2)*

<표 1>은 관계형 데이터베이스에서 SQL을 사용하여 클래스 다이어그램을 검색하는 방법을 순차적으로 나타낸다. 먼저, 클래스 다이어그램의 클래스들을 검색한다. 다음, 각 클래스 내부의 애트리뷰트들과 오퍼레이션들을 검색한다. 마지막으로, 각 클래스 간의 관계들을 검색함으로써 원하는 클래스 다이어그램을 전체적으로 검색할 수 있다.

4. 결론 및 추후 연구과제

UML은 기존의 다양한 시스템 개발 방법론들을 하나의 틀로 통합한 것이다. 특히, 소프트웨어 시스템의 설계 및 개발 등을 체계적으로 지원하는 모델링 언어이다. 이러한 UML로 개발된 모델들의 효율적인 사용을 위하여 통합하여 저장하고 관리하는 것이 필요하다.

본 논문에서는 UML 클래스 다이어그램을 관계형 데이터베이스로 사상시키는 알고리즘과 질의하는 알고리즘을 제안하였다. 제안한 알고리즘에 따라 설계 정보를 데이터베이스화함으로서 변화하고 있는 컴퓨팅 환경과 개발 방법론에 빠르게 대처할 수 있다. 또한 다수의 사용자가 설계에 참여할 수 있도록 공유함으로써 설계과정의 효율을 향상시키는 데도 기여할 수 있을 것이다.

본 연구의 추후 과제로는 UML로 표기된 모든 종류의 다이어그램을 관계형 데이터베이스에 저장하는 것과 클래스 다이어그램에 순공학 및 역공학 기법을 도입하여 설계가 곧 프로그래밍과 연결될 수 있도록 하는 것이다. 또한, 분산 환경에서의 소프트웨어 개발을 지원할 수 있도록 하는 것이다.

참 고 문 헌

[1] http://www.omg.org - OMG home page, contains specification for UML and related modeling standards, such as MOF and XMI.

[2] Booch, Rumbaugh, Jacobson, The Unified Modeling Language User Guide, Addison Wesley, 1997.

[3] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, Database System Concepts, 3th, McGraw-Hill, 1996.

[4] 강설문, 김태희, "객체지향 소프트웨어 개발 방법론의 표준화: UML(Unified Modeling Language)", 정보처리논문지, 제 5권 제5호, 1998. 8.