# UML 설계 정보 분석을 위한 규칙 기반 항해 정보 표현 방안

김원중* 배명남** 양재동* 양현택***
*전북대학교 컴퓨터과학과
**한국전자통신연구원 실시간 DBMS팀
***순천대학교 컴퓨터과학과

# A Representation Scheme of Rule-Based Navigation Information for UML Design Object Analysis

Won-Jung Kim* Myung-Nam Bae** Jae-Dong Yang* Hyun-Teak Yang***
*Dept. of Computer Science, Chonbuk University
**Real-time DBMS Department ETRI
***Dept. of Computer Science, Sunchon University

**요 약**

본 논문에서는 UML로 작성한 설계요소들간의 관련성을 통해 설계내역을 정확하고 쉽게 분석하기 위한 방법을 제안한다. 이 방법은 1) 설계요소들이 갖는 관련성을 위한 표현 규칙을 정의하고, 2) 정의된 규칙에 따라 관련 있는 설계요소들에 관련성을 부여하며, 3) 이 관련된 설계요소들을 쉽게 검색하고 항해할 수 있는 수단을 제공한다. 제안된 방법을 통해, 보다 정확하고 체계적인 UML 설계요소들간의 분석이 가능하다.

## 1. 서론

UML은 소프트웨어 개발자에게 표현력이 강한 시각적 모델링 언어를 제공함으로써 의미 있는 모델들을 개발하고 서로 교환할 수 있도록 도와준다. 시스템 분석가나 개발자는 UML을 사용한 소프트웨어 개발 동안 생성되는 많은 설계내역을 통해 시스템의 구조 및 시스템의 흐름을 자연스럽게 파악할 수 있다.

그러나, UML 설계 도구들은 설계된 설계내역들간의 관련성 인식이 부족한 중요한 단점이 있다. 예를 들어, 클래스의 특정 메소드가 수행된 후, 이 메소드의 실행에 관련된 모든 클래스들을 검색하고 싶은 경우, UML 설계 도구는 이러한 정보를 제공할 수 없기 때문에 관련 있는 여러 설계요소들간의 의미 파악이 어렵다. 그러나, 이들에게 미리 정의한 관련성을 부여한다면, 이를 사용하여 설계내역이 가지는 정보의 흐름, 상호 작용하는 관계, 그리고 실행 과정 추적 등의 특성을 일목요연하게 파악할 수 있다.

본 논문은 UML 설계 도구에 설계요소들간의 유용한 관계를 정의하고, 생성된 설계내역 및 설계요소들간 관계를 부여하고, 실제 개발 환경에서 여러 설계정보들간의 관련 정보를 추적할 수 있는 항해 방법을 설명한다.

본 논문의 구성은 다음과 같다. 먼저, 2장 관련연구에서는 UML 및 이를 지원하는 설계도구와 관련된 두 가지 사례를 기술하며, 3장에서는 제안하는 설계구조 및 내용을 설명한다. 4장에서는 설계내역과 설계요소들간의 의미적인 관계성을 부여하기 위한 예를 보이며, 5장에서는 검색을 위해 제시된 질의 내용과 예를 보인다. 마지막으로 6장에서는 결론 및 향후 연구 과제를 제시한다

## 2. 관련연구

### 2.1 UML

UML은 객체지향 분석(Analysis)과 설계(Design)를 위한 모델링 언어이며, 시스템의 모든 객체와 그들 사이의 관계성을 그래픽 표기로 모델링함으로써 유지보수가 용이하고, 재사용성이 높은 소프트웨어를 생산할 수 있다[2,3,5].

UML은 기본적으로 여러 개의 다이어그램을 제공한다. 대표적인 다이어그램으로 클래스 다이어그램은 시스템의 정적이고도 구조적인 특성을 기술하는데 사용하며, 시퀀스 다이어그램은 시스템의 동적인 구조를 기술하는데 사용한다. 즉, 동적인 행위의 흐름에 따라 객체들 사이의 상호작용을 표현한다. 이외에도 유스케이스, 오브젝트, 콜라보레이션, 스테이트, 액티버티, 컴포넌트, 디플로이먼트 다이어그램 등이 있다. 이 다이어그램들은 하나의 객체를 고유의 기술 양식에 따라 그 특성을 기술한다[2,6].

### 2.2 UML을 지원하는 설계 도구

Rational Rose[4]는 트리 구조를 통해 설계요소들에 대한 참조 구조를 제공한다. 그러나, Rose는 UML이 제공하는 노테이션(notation)이 갖는 문법적인 관계에서 추출된 정보만을 제공한다. 예를 들어, 설계요소들 중 'parking()' 메소드를 갖는 'Car' 클래스와 'forward()' 메소드를 갖는 'Gear' 클래스가 'parking()'과 'forward()'를 통해 관계를 갖는다고 할 때, 'Car'와 관련된 'Gear'를 검색하기 어렵다. 즉, 노테이션이 갖는 표현 규칙만으로 파악할 수 없는 정보의 흐름도 극히 제한적이다.

Argo/UML[1]은 Rational Rose가 제공하는 노테이션의 표현 정보만을 사용한 설계요소의 단점을 일부 해결하고 있다. 즉, 노테이션이 지니는 문법적인 정보를 활용하여 설계요소들간의 관련성을 표현하기 위한 일련의 규칙들을 제공한다. 예를 들어, 규칙들 중 class->navigable class를 통해 'Car'와 'Gear'가 관련성이 있음을 인식할 수 있다. 그러나, 이 규칙들은 의미적인 관련성을 통해 정보의 흐름을 파악할 수 있는 설계요소간 항해를 제공하지 못하는 제약이 있다.

## 3. 설계요소간의 의미 정보 파악과 저장 구조

소프트웨어 전 개발 과정에서 생성되는 설계내역은 설계요소들의 집합으로 구성되며, 이러한 설계요소들간의 의미적인 관련성은 시스템의 전체 흐름을 파악하는데 매우 중요하다.

예를 들어, 그림 1을 통해 정보의 흐름을 파악해 보자.

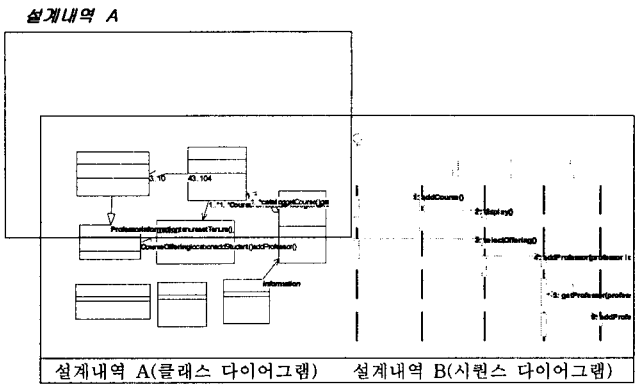


그림 1 설계내역의 예

설계내역 A의 'CourseForm' 클래스에 속하는 'selectOffering()' 메소드 수행시에 필요한 클래스들을 검색하고자 할 경우, 클래스 다이어그램에 속하는 'CourseForm' 만으로 'Course'와 관련성이 있다는 사실을 파악하기는 어렵다. 그러나, 설계내역 B의 ':CourseForm'는 'CourseForm'의 인스턴스이고, ':Course'는 'Course'의 인스턴스이다. 또한, ':CourseForm'에 속하는 'selectOffering()' 메시지 실행 후 순차적으로 ':Course'에 속하는 'addProfessor(professor id)' 메시지를 호출하는 관계를 통해 'selectOffering()' 메소드 수행시에 필요한 클래스들 중 하나가 'Course'임을 알 수 있다.

본 논문에서는 의미적인 관계를 갖는 설계요소들을 파악하기 위해 요소들간의 검색 및 항해 방법을 제안하고자 한다. 개략적인 시스템의 구조는 아래와 같다.

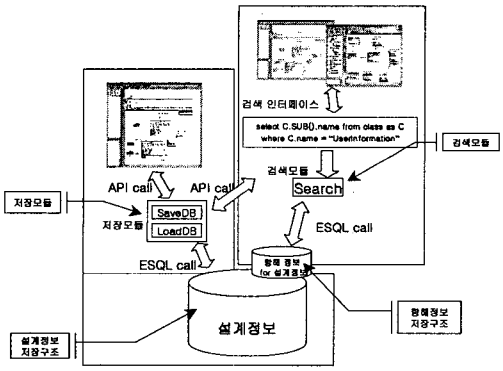


그림 2 저장 구조

먼저, 항해정보 저장구조는 생성되는 설계요소가 다른 설계요소들과 관련성이 있을 때, 해당하는 관련 정보를 저장한다. 이 관련 정보는 설계요소들간의 관련성을 파악할 때, 설계요소로의 참조를 위해 사용된다. 저장모듈은 설계도구에 의해 생성된 모든 설계요소들을 설계정보 저장 구조에 저장한다.

여기에서, 저장된 설계정보를 토대로 시스템의 흐름을 파악할 경우, 검색 인터페이스는 사용자가 원하는 질의를 얻는다. 또한, 검색모듈은 주어진 질의를 통해 항해정보로부터 관련 정보를 추출한다. 이 정보는 사용자의 요구에 맞는 설계요소를 설계정보로부터 추출하기 위한 수단으로 제공된다.

## 4. 관련성 정보의 표현

이 절에서는 의미적인 관계가 있는 설계요소들간의 관련성을 표현하기 위한 모델을 제시한다. 설계요소들이 갖는 관계들 중 의미적으로 관련된 요소들을 UML 설계 도구들이 제공하는 노테이션만으로 표현하는 것은 어렵다. 따라서, 설계요소들간의 의미적인 관계를 표현하기 위한 관련성 정보가 필요하다. 관련성을 통한 설계요소들간의 정보의 흐름은 시스템 전체 구조를 쉽게 파악할 수 있도록 한다.

관련성 정보는 추상화를 통해 설계요소가 지니는 속성들을 쉽게 나타낼 수 있으며, 정보 접근을 위해 질의어를 사용하기 때문에 복잡성이 감소된다.

본 논문에서 설계요소들간의 관련성을 표현할 때, 설계요소는 노드로, 관계는 이름을 라벨로 표현한다. 또한 추상화 모델은 모든 설계내역과 설계요소들을 유일한 식별자로 갖게되며, 이 식별자는 전 설계 과정에서 한 번만 생성되고, 재사용 되지 않는다. 관계를 갖는 설계요소들간의 참조는 모두 이 식별자를 통해 이루어진다.

설계요소간의 관련성 = ($ID_i$, R, $ID_j$)
$ID_i$ = 설계요소의 식별자
R = 유향 관계의 종류
$ID_j$ = $ID_i$와 R이 명시한 관계를 갖는 설계요소의 식별자

예를 들어, 'Car' 클래스와 'Gear' 클래스가 연관 관계가 있을 때, 'association'을 다음과 같이 표현한다.

설계요소간의 관련성 = (700, out_ass, 720)
700 = 'Car'의 식별자
out_ass = 외향 연관 관계
720 = 'Car'와 'out_ass'이 명시한 관계를 갖는 'association'의 식별자

### 4.1 의미 정보 표현

다음의 그림 3은 설계요소들중 클래스, 오브젝트, 액터, 그리고 유스케이스 등의 관련성을 표현할 때, 설계요소는 노드로, 관계는 이름을 라벨로 표현하는 항해 모델을 보여주고 있다.

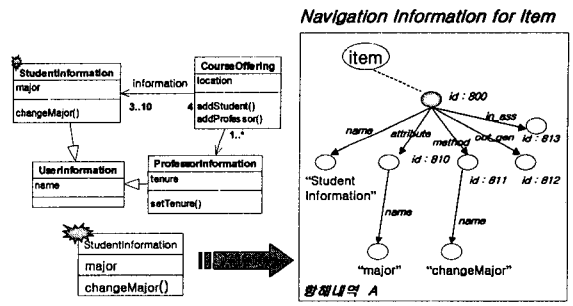


그림 3 ITEM 관련성 정보 표현

설계내역 A에서 'StudentInformation' 클래스는 'UserInformation' 클래스의 서브클래스이고, 'CourseOffering' 클래스와 연관 관계에 있다. 또한, 'major' 애트리뷰트, 'changeMajor()' 메소드로 구성된다. 즉, 'StudentInformation' 클래스가 갖는 관련성 정보는 항해내역 A와 같이 그래프 방식으로 표현된다.

설계요소 중 관계들로는 '연관(association)', '집합(aggregation)', '종속(dependency)', 그리고 '상속(generalization)' 등이 있다.

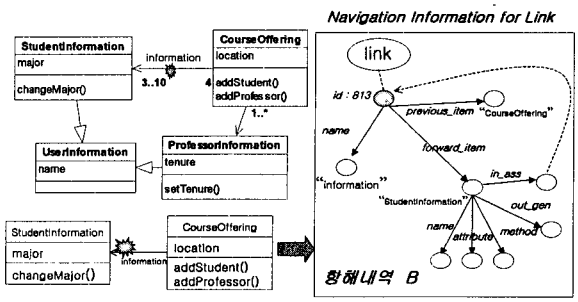


그림 4 LINK 관련성 정보 표현

예를 들어, 그림 4에서 'information' 연관은 'StudentInformation' 클래스와 'CourseOffering' 클래스간의 관계를 설정하고 있다. 이때, 'information'은 'CourseOffering' 클래스의 메소드에 의해 호출되는 클래스를 명시하기 위한 연관 관계의 이름이고, 연관 정보는 'previous_item', 'forward_item' 등이 있다.

### 4.2 설계내역내 정보 표현의 예

설계요소들이 갖는 관련성 정보는 미리 정의한 질의로 결과를 얻게 된다. 이런 경우, 내부적으로 요소간의 관련성에 따라 자유롭게 항해하기 위해 정의한 규칙에 의해 정보를 추출할 수 있다. 관련성 정보를 파악하기 위한 규칙메소드와 규칙의 예는 다음과 같다.

규칙메소드 : $F_{\text{규칙메소드명}}$(객체식별자) → 객체식별자집합

규칙메소드는 모델내에서 노드간의 항해를 위해 정의하였다. 예를 들어, 클래스 $C_1$의 하위클래스들의 객체식별자는 $C_1$.in_gen.previous_item으로 얻을 수 있으며, 적용될 규칙메소드는 아래와 같다.

$F_{GETSUB}$(객체식별자) : 객체식별자집합

여기에서, GETSUB에 의해 수행되는 $C_1$.in_gen.previous_item을 살펴보면, '.'은 명시한 관련성(in_gen과 같은)과 관련된 새로운 설계요소를 지칭하기 위해 사용된다. 'in_gen'은 일반화를 나타내는 설계요소의 식별자를 얻고, 다시 'previous_item'을 사용하여 얻어진 식별자로 하위클래스관계로 명시된 설계요소들의 식별자를 얻을 수 있다. 규칙은 규칙메소드를 조합하여 보다 사용자에게 친숙한 규칙을 구성하는데 사용된다.

모델내에 가능한 모든 객체식별자 집합이 M이라 할 때, 객체식별자 $\{o_1,o_2,o_3,...\}$ = M에 적용되는 규칙 R의 정의는 다음과 같다.

$R_{\text{규칙명}}(\{o_1,o_2,o_3,...\})$= M′∪ $R_{\text{규칙명}}$(M′), M′={o′| o′∈ $F_{\text{규칙메소드명}}(o_1) \vee o' \in F_{\text{규칙메소드명}}(o_2) \vee o' \in F_{\text{규칙메소드명}}(o_3) \vee ...$)

여기에서, $\{o_1,o_2,o_3,...\}$와 M′의 객체식별자를 갖는 설계요소는 규칙명을 파악할 수 있는 속성을 갖는다. 이를 다시 정의하면, 다음과 같다.

규칙 : DEFINE <규칙명>
AS <규칙메소드>
BASED ON <객체식별자>

예를 들어, $C_1$에서 상속받은 모든 하위클래스를 얻는 규칙 SUB는 아래와 같이 정의할 수 있다.

DEFINE SUB AS GETSUB
BASED ON <$C_1$의 객체식별자>

정의된 SUB 규칙을 규칙메소드를 가지고 표현을 하면 아래와 같다.

$R_{SUB}(\{o\})$ = M′ ∪ $R_{SUB}$(M′)
= $\{F_{GETSUB}(o)\} \cup R_{SUB}(\{F_{GETSUB}(o)\})$
= $\{o_1,o_2,o_3,...\} \cup \{F_{GETSUB}(o_1),F_{GETSUB}(o_2),F_{GETSUB}(o_3),,,...\}$
$\cup R_{SUB}(\{F_{GETSUB}(o_1),F_{GETSUB}(o_2),F_{GETSUB}(o_3),...\})$

그림 5는 'Shape' 클래스의 관련성 정보에 관한 내용을 그래프로 표현한 것이다.

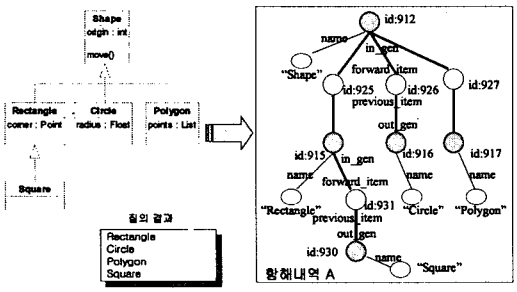


그림 5 'SUB' 관련성 정보 표현

예를 들어, 설계요소인 'Shape' 클래스로부터 상속받는 클래스들을 검색하고자 할 때, 우선, 'Rectangle', 'Circle', 그리고 'Polygon' 클래스가 추출될 것이다. 그리고 'Rectangle' 의 하위 클래스인 'Square' 클래스가 추출된다. 즉, SUB 규칙을 통해 다음과 같이 처리된다.

$R_{SUB}(\{912\})$ = $\{F_{GETSUB}(912)\} \cup R_{SUB}(\{F_{GETSUB}(912)\})$
= {915, 916, 917}
∪ $\{F_{GETSUB}(915),F_{GETSUB}(916),F_{GETSUB}(917)\}$
∪ $R_{SUB}(\{F_{GETSUB}(915),F_{GETSUB}(916),F_{GETSUB}(917)\})$
= {915, 916, 917}∪{930}∪$R_{SUB}(\{F_{GETSUB}(930)\})$
= {915, 916, 917, 930}

## 5. 질의어

이 절에서는 제안하는 모델을 기초로 한 질의어를 설명한다. 질의어를 설계하기 위하여 데이터 질의어의 표준인 SQL을 확장하여 활용하는 방법을 고려하였다. 즉, 기존의 SQL 언어로 표현이 불가능한 부분은 그래프로 표현된 관련성 정보 모델에 맞게 재구성하여 표현하고자 하였다. 이 방법은 관련성을 표현하기 위해 사용한 규칙과 정보를 이용하여 질의를 할 수 있도록 하였다.

질의는 임의의 설계요소와 관련성을 갖는 설계요소를 파악하고자 할 때 사용한다. 예를 들면, 'Shape' 클래스와 상속 관계를 갖고, 서브클래스에 해당하는 관련성을 갖는 다른 설계요소로 항해하고자 할 때 다음과 같은 질의가 가능하다.

사용자 질의:
SELECT C.SUB().name
FROM class_repository as C
WHERE C.name = "Shape"

여기에서, 'SUB()'는 앞서 정의한 규칙에 의해 'Shape' 클래스와 일반화관계를 갖는 하위클래스의 추출을 명시하고 있다. 구체적으로, 클래스 이름이 'Shape'인 조건절의 명시를 통해 항해하려는 설계요소를 통해 클래스로 제한된다. 그 결과 'C'에 정의된 규칙 'SUB'를 통해 검색한다. 그리고, 검색된 클래스의 'name'을 검색한다. 'C.SUB'의 경우에 검색된 내용은 여러 개 일 수 있다. 즉, 설계요소는 일대일의 관계만을 갖는 것이 아니라 일대다의 관계를 갖기 때문이다. 실제적인 질의 처리는 아래와 같이 규칙과 규칙메소드들의 조합으로 이루어진다.

$t1 = F_{GETOID}$("Shape")
$t2 \in R_{SUB}(t1)$
printf("%s",$F_{GETNAME}(t2)$)

즉, $F_{GETOID}$()는 'Shape' 클래스의 식별자 '912'를 얻는다. 다음으로, $t_1$('912')을 인자로 갖는 규칙 SUB를 통해 '915', '916', 그리고 '917'을 얻는다. 즉, t2는 t1에 의해 얻어지는 객체 식별자이다. 마지막으로, t2의 'name' 속성에 의해 'Shape'의 하위클래스 이름들이 추출된다.

앞서 정의한 규칙과 규칙메소드들은 이러한 SQL 수준의 질의를 통해 UML로 작성된 설계내역내의 관련 설계요소들을 쉽게 추출할 수 있다.

## 6. 결론 및 향후 연구 과제

UML은 소프트웨어의 분석, 설계, 구현의 전 개발주기를 잘 지원하는 매우 효과적인 개발 방법론이며, 객체 지향 개발 방법론의 대표적인 것이라 말할 수 있다[2,3,5].

본 논문에서는 이러한 UML 설계 도구에서 생성되는 설계 요소들간의 상호 관련성을 통해 시스템의 구조, 정보의 흐름을 파악할 수 있는 방법론에 대해 기술하였다. 의미 정보 파악을 위해 규칙을 정의하고, 사용자의 이해를 돕기 위해 상호 관련성의 의미적인 내용을 그래프로 표현하였다. 또한, 관련성 정보를 검색하기 위해, 데이터 질의어의 표준인 SQL을 연관성 모델에 반영하도록 변경, 확장하여 활용한 질의어를 정의하였다. 이를 통해, 소프트웨어 개발중에 얻은 설계요소들간의 연관성을 파악하고 설계내역간의 항해를 제공함으로써 효과적으로 소프트웨어의 분석과 시스템의 전체 흐름을 이해하는데 도움을 줄 수 있다.

계속적으로 이루어져야 할 연구 과제로서, 1) 개발된 설계 도구와의 통합 및 검증에 대한 연구, 2) 모델에 보다 적합한 질의어 개발등을 들 수 있다.

### 참고 문헌

[1] Argo/UML v0.7: The Cognitive CASE Tool, http://argouml.tigris.org/, University of California, Irvine, 1999.
[2] Booch, G., Rumbaugh, J., and Jacobson, I. 1999, The Unified Modeling Language User Guide, Addison- Wesley Publication Company.
[3] Grady Booch, Ivar Jacobson, and James Rumbaugh. Unified Modeling Language. Rational Software Corporation. January 1997. Version 1.0.
[4] Rational Soft. Corp, http://www.rational.co.kr/Product/Rose/.
[5] Rational Soft. Corp, "Unified Modeling Language," http://www.rational.com/uml, 2000.
[6] UML 1.3 Specification. OMG Documents ad990608-ad990609.