Journal of the Research Institute of Industrial Technology, Vol. 18, 2001

UML 클래스 다이어그램의 관리를 위한 관계형 데이터베이스 설계 및 구현

이 성 대[1], 박 휴 찬[2]

Design and Implementation of a Relational Database for Management of UML Class Diagrams

S. D. Lee , H. C. Park

요 약

UML (Unified Modeling Language)은 국제표준기구인 OMG (Object Management Group)에서 표준으로 지정한 통합된 시스템 개발방법론으로서 시스템의 설계 및 개발 등을 체계적으로 지원하며, 시스템을 가시화(visualizing), 명세화(specifying), 구조화(constructing) 그리고 문서화(documenting)하는 모델링 언어이다. 이러한 UML을 이용하여 개발된 모델들을 효율적으로 관리하기 위하여 통합된 시스템의 개발이 필요하다. 이를 위하여 본 논문에서는 UML 모델을 저장하고 검색하기 위한 관계형 데이터베이스(Relational Database)를 설계하고, 설계된 데이터베이스로부터 사용자가 원하는 UML 모델을 검색하는 방법을 제안한다. 제안한 방법을 구현한 시스템을 사용함으로써 모델 정보를 효율적으로 관리할 수 있고 다수의 사용자가 서로 공유할 수 있어 모델의 재사용(reuse)이 가능할 것이다.

1. 서 론

최근 소프트웨어가 고부가가치 상품으로 인식되고 있고, 업무 처리 과정에서 소프트웨어가 처리할 업무의 범위와 규모가 커짐에 따라 시스템의 복잡성을 체계적으로 관리할

1) 한국해양대학교 대학원
2) 한국해양대학교 기계·정보공학부

필요성이 증가되고 있다. 따라서 소프트웨어 생산을 자동화하고 개발에 필요한 시간과 비용을 절감하여 소프트웨어의 질을 향상시킬 수 있는 기술들이 모색되고 있는 데, 이 중에서 개발할 시스템의 구성 요소를 체계적으로 분석하고 설계할 필요성에 따라 제안된 것이 UML (Unified Modeling Language)이다.

UML은 분산 객체 어플리케이션 개발을 위한 공통 프레임워크 지침(framework guideline)과 상세한 객체 관리규약의 정립을 목표로 1989년 설립된 OMG (Object Management Group)에서 시스템 분석 및 설계 방법론의 표준 지정을 목표로 제안하였다. UML은 Rumbaugh의 OMT 방법론[1], Booch의 Booch 방법론[2,3] 그리고 Jacobson의 OOSE 방법론[4] 등 다수의 객체지향 방법론들을 통합하여 만든 통합 모델링 언어이며, 이미 많은 시스템 개발 도구들이 UML을 지원하고 있다.

본 논문에서는 다수의 사용자가 UML을 사용하여 설계한 모델을 공유하여 사용할 수 있도록 하기 위하여 관계형 데이터베이스에 이 모델들을 저장하고 검색하는 방법을 제안한다. UML의 구성요소 중에서 가장 핵심이 되는 부분이 클래스 다이어그램(Class Diagram)이다. 이 클래스 다이어그램을 저장하기 위해서 먼저 클래스 다이어그램을 구성하고 있는 다이어그램(diagram), 클래스(class), 속성(attribute), 연산(operation) 그리고 관계(relationship)를 저장할 수 있는 테이블을 설계하고, 이 테이블을 이용하여 저장하고 검색하는 방법을 제안한다. 제안한 방법을 적용하여 구현한 시스템은 UML 클래스 다이어그램의 구성요소들을 각각의 테이블로 구성하고, 이 테이블들을 서로 연결하기 위해 각 테이블간의 관계를 설정하여 계층적인 구조를 저장할 수 있도록 한다. 테이블간의 관계에는 기본키(primary key)와 참조키(foreign key)가 있다.

본 논문의 2장에서는 UML의 구성요소와 클래스 다이어그램에 대해 간략하게 설명하고, 3장에서는 제안하는 UML 클래스 다이어그램을 저장하고 검색하기 위한 관계형 데이터베이스 테이블을 설계하고, 4장에서는 시스템 구현을 위한 저장 및 검색 알고리즘과 전체적인 시스템의 흐름도에 대해서 설명하고, 기존 시스템과 비교한다. 5장에서는 결론과 함께 추후 연구과제에 대해서 논의한다.

2. UML 및 UML 개발 도구

UML은 통합된 시스템 개발방법론으로서 시스템을 가시화, 명세화, 구조화 그리고 문서화하는 것이다. 특히 기존에 존재하던 여러 개발 방법론[1,2,3,4]들의 장점들을 모두 수용했으며, 확장이 용이하고, 다양한 표기법을 가지고 있어 실시간 시스템뿐 아니라 여러 종류의 시스템 개발에 사용되어 질 수 있다.

2.1 UML의 구성 요소

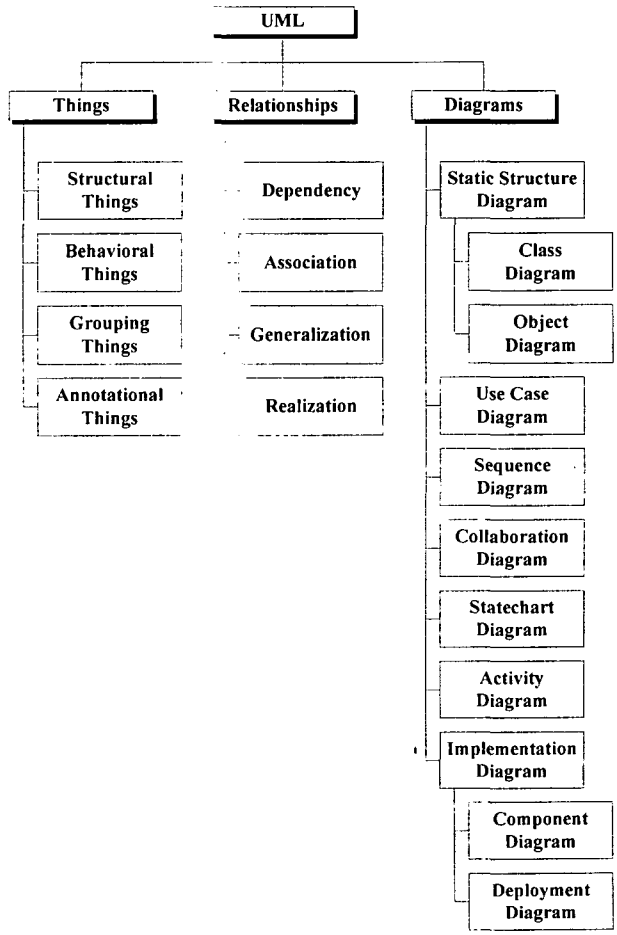


그림 1. UML의 구성 요소

UML은 그림 1과 같이 크게 사물(Things), 관계(Relationships), 다이어그램(Diagrams)의 3가지 구성요소로 구분할 수 있다. 첫 번째 구성요소인 사물은 UML 모델의 실체를 나타내는 구조 사물(Structural Things), 동적인 부분을 나타내는 행동 사물(Behavioral Things), 모델의 조직적인 부분을 나타내는 그룹 사물(Grouping Things), 그리고 모델에 대한 추가적인 설명을 표현하는 주해 사물(Annotation Things)로 나눌 수 있다. 두 번째 구성요소는 관계로서 의존(Dependency), 연관(Association), 일반화(Generalization), 실제화(Realization) 그리고 연관 관계의 확장 형태인 집합 연관(Aggregation) 관계로 나눌 수 있다. 마지막 구성요소로는 설계를 시각화하는 다이어그

램이다. 다이어그램을 세분화하면 정적 구조(Static Structure), 사용 사례(Use Case), 순차(Sequence), 협력(Collaboration), 상태도표(Statechart), 활동(Activity) 그리고 구현(Implementation) 다이어그램이 있다. 다시, 정적 구조(Static Structure) 다이어그램은 클래스(Class), 객체(Object) 다이어그램으로, 구현(Implementation) 다이어그램은 컴포넌트(Component), 배치(Deployment) 다이어그램으로 세분할 수 있다[5,6,10,11,12].

2.2 UML 클래스 다이어그램(Class Diagram)

UML의 많은 다이어그램들 중에서 가장 중심이 되는 것은 정적 구조 다이어그램으로 시스템을 논리적인 관점에서 분석한다. 여기에는 클래스 다이어그램과 객체 다이어그램이 있으며, 클래스 다이어그램은 클래스, 인터페이스(interface)와 같은 모델 요소와 모델 요소 간의 관계들로 구성되어 있으며, 정적인 구조 모델을 표현하는 다이어그램이다. 객체 다이어그램은 클래스 다이어그램의 인스턴스(instance)이다[5,6,10].

그림 2는 클래스 다이어그램의 일반적인 구조를 나타내고 있다. 하나의 클래스 다이어그램은 내부에 클래스와 관계로 구성이 되고, 클래스는 클래스 이름(name)과 속성 목록(attribute list) 그리고 연산 목록(operation list)으로 구성이 된다.

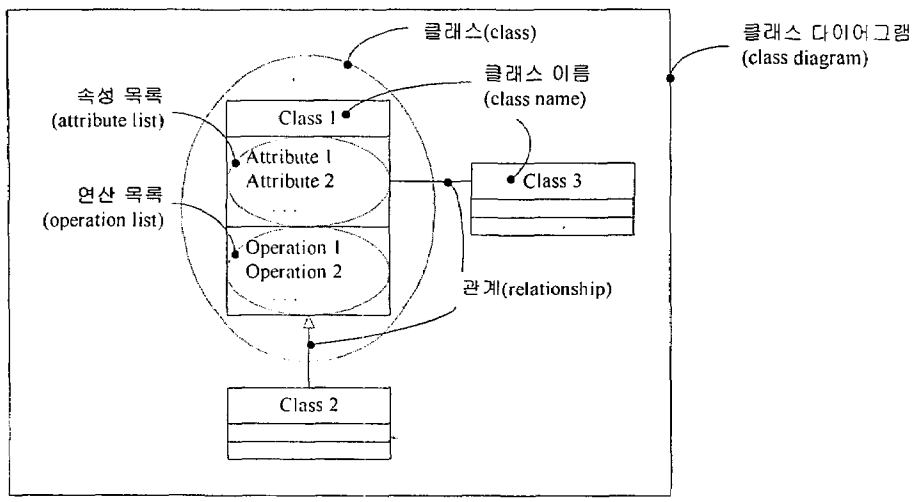


그림 2. UML 클래스 다이어그램의 구조

클래스는 비슷한 속성과 공통의 행위를 갖는 객체들의 그룹을 기술하는 단위로서 사물을 추상화하고 단순화하는 것이다[5,6,10]. UML에서는 그림 3과 같이 클래스를 표현하며, 계층적인 구조를 가지고 있다. 각각의 속성이나 연산은 가시성(visibility) 표시를

가지는 데 '+'는 public, '-'는 private, '#'은 protected 속성을 나타낸다.

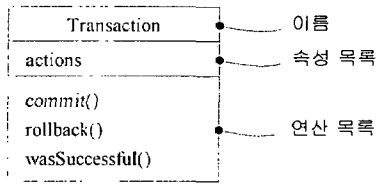


그림 3. 클래스의 표현

인터페이스는 연산(operation)들의 모임으로 하나의 클래스나 컴포넌트들이 제공하며, 서비스를 명세화하기 위해 사용한다. 인터페이스는 그림 4의 (a)와 같이 원을 사용하여 단지 인터페이스임을 명시하는 방법과 그림 4의 (b)와 같이 클래스의 스테레오타입(stereotype)에서 인터페이스를 명시하는 방법이 있다.

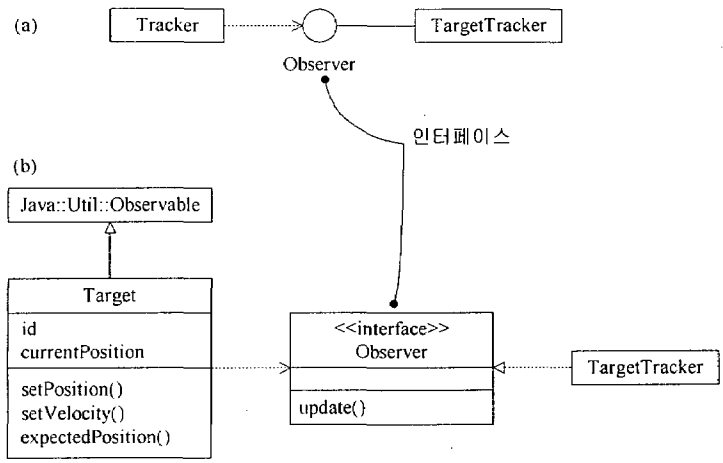


그림 4. 인터페이스의 표현

관계는 클래스와 클래스, 클래스와 인터페이스의 상호 연관성을 나타낸다. 관계에는 의존(dependency), 일반화(generalization), 연관(association), 집합 연관(aggregation) 그리고 실제화(realization) 관계가 있다. 의존 관계는 사용 관계로서 한 사물의 명세서가 바뀌면 이를 사용하는 다른 사물에게 영향을 끼치는 것을 의미한다. 즉, 사용 중인 클래스가 바뀌면 상대 클래스의 연산(operation)이 함께 영향을 받는다. 일반화 관계는 "is-a" 또는 "kind-of"의 관계로서 상위 클래스의 속성(attribute)과 연산(operation)이 하

위 클래스로 상속되는 것을 의미한다. 연관 관계는 "has-a" 관계를 나타내며 구조적인 관계로서 특정 사물 객체가 다른 사물 객체와 관계가 있음을 표현하고, 역할(role)과 다중성(multiplicity)을 표시할 수 있다. 집합 연관 관계는 연관 관계의 확장형태로서 "part-of"의 관계이다. 실제화 관계는 인터페이스나 컴포넌트(component)와의 상호 연관성을 표시하는 관계이다. 그림 5에서 클래스 "Window"는 클래스 "Event"와 의존 관계를 지니고 있으며, 클래스 "ConsoleWindow"와 "DialogBox"는 클래스 "Window"로부터 일반화, 즉 상속되었음을 나타낸다. "Control" 클래스는 "DialogBox" 클래스와 연관 관계를 지니고 있음을 나타낸다.

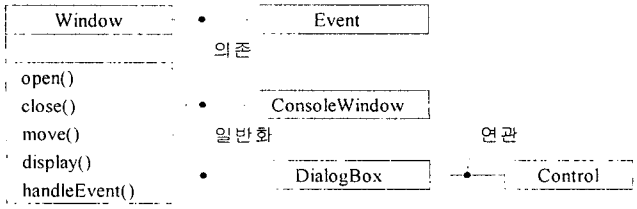


그림 5. 관계의 표현

UML을 지원하는 대표적인 시스템은 UML을 제안한 Booch, Rambaugh와 Jacobson이 중심이 되어 개발한 "Rose"와 국내의 "Plastic"이 있다. Rose에서 모델 정보를 저장하는 방법은 파일 저장 시스템을 이용한다. UML의 모든 다이어그램을 지원하며, 다이어그램을 C++, Java 등과 같은 객체 지향 언어의 코드를 생성시키는 순공학(forward engineering)과 객체를 표현하고 있는 코드에서 다이어그램을 생성하는 역공학(reverse engineering) 기능을 지원한다[6]. Plastic은 국내에서는 UML을 지원하는 유일한 도구로서 모델 정보의 저장은 파일 시스템을 사용하며 지원하는 다이어그램의 종류는 객체(Object) 다이어그램을 제외한 총 8가지이다. 자바(java) 코드에 대한 순공학과 역공학을 지원한다[7].

3. UML 클래스 다이어그램의 저장 및 검색

UML 클래스 다이어그램은 다수의 클래스와 관계들로 구성되며, 각 클래스 내부에는 다수의 속성과 연산이 존재하는 계층적인 구조를 지니고 있다. 관계형 데이터베이스에서는 이러한 계층적인 구조를 허용하지 않으므로 클래스 다이어그램을 저장하기 위한 관계형 데이터베이스 테이블을 설계한다. 또한, 이 데이터베이스에서 클래스 다이어그램을 검색하는 방법에 대해서 논의한다.

3.1 관계형 데이터베이스 설계

이름	타입	설명
Diagram_ID	int	P-K
Diagram_Name	varchar	이름
Diagram_Info	varchar	기타 정보
Diagram_Owner	char	F-K
Diagram_FDate	datetime	현재 일자
Allow_Others	bit	공개 여부

(a) Diagram

이름	타입	설명
Class_ID	int	P-K
Class_Name	varchar	이름
Class_Info	varchar	기타 정보
Diagram_ID	int	F-K
Is_Interface	bit	인터페이스
Stereotype	varchar	스테레오타입
Start_X	int	시작 X, Y 좌표
Start_Y	int	

(b) Class

이름	타입	설명
Attr_ID	int	P-K
Attr_Name	varchar	이름
Attr_Visibility	int	가시성 번호
Attr_Type	varchar	타입
Attr_Stereotype	varchar	스테레오타입
Class_ID	int	F-K
Attr_Info	varchar	기타 정보

(c) Attribute

이름	타입	설명
Oper_ID	int	P-K
Oper_Name	varchar	이름
Oper_Type	varchar	반환 타입
Oper_Para	varchar	인자 리스트
Oper_Visibility	int	가시성 번호
Oper_stereotype	varchar	스테레오타입
Class_ID	int	F-K
Oper_Info	varchar	기타 정보

(d) Operation

이름	타입	설명
Relationship_ID	int	P-K
Super_Class	int	F-K
Sub_Class	int	F-K
Relationship	int	관계 번호
Relationship_Info	varchar	기타 정보
Super_multi	char	다중성 표시
Sub_multi	char	
Super_Rolename	varchar	역할명 표시
Sub_Rolename	varchar	

(e) Relationship

그림 6. 클래스 다이어그램의 저장과 검색을 위한 데이터베이스 테이블 구조

그림 2와 같은 구조를 지닌 클래스 다이어그램을 각각의 구성요소들의 특징을 고려하여 그림 6과 같은 데이터베이스 테이블을 설계한다. 그림 6의 (a)는 클래스 다이어그램을 저장하기 위한 테이블로서, 클래스 다이어그램을 작성한 사용자(Diagram_Owner)와 공개 유무(Allow_Others)를 갖는 필드가 있다. 사용자는 다이어그램을 검색할 경우 자신이 작성한 모델과 다른 사람이 작성한 공개 모델을 검색할 수 있다. 또한 사용자가 임의로 입력하는 정보(Diagram_Info)를 추가함으로써 이후 검색의 효율성을 높일 수 있도록 하였다. 그림 6의 (b)는 클래스의 정보를 저장하는 테이블이다. 소속 다이어그램을 참조키(foreign key)로 지정하여 잘못된 데이터의 입력을 미연에 방지한다. 클래스를 검색한 후 화면 출력을 위하여 시작 좌표(StartX, StartY)와 사용자가 입력하는 클래스 정보(Class_Info)를 저장함으로서 클래스 이름 외의 기타 정보를 사용하여 검색할 수 있도록 한다. 그림 6의 (c)와 (d)는 클래스 내부의 속성과 연산을 저장하는 테이블로서 그림 6의 (b) Class 테이블의 기본키(primary key)인 Class_ID를 참조하여 생성된다. 그림 6의 (e)는 클래스나 인터페이스와 같은 모델 요소 간의 관계를 저장하는 테이블로서, 관계와 연결된 클래스의 다중성(multiplicity)이나 역할명(role name)은 연관과 집합 연관 관계인 경우에 사용된다.

3.2 UML 클래스 다이어그램 검색

임의의 클래스 다이어그램의 이름이 주어졌을 때, 데이터베이스에서 SQL(Structured Query Language)을 사용하여 클래스 다이어그램을 검색한다. 데이터베이스의 각 테이블로부터 클래스 다이어그램을 검색하기 위해서는 다이어그램 테이블에서 사용자가 요구하는 다이어그램의 식별자를 먼저 추출한다. 추출된 클래스 다이어그램의 식별자를 검색어로 사용하여 소속된 클래스와 인터페이스들을 검색한다. 다음은 각 클래스 내부의 속성들과 연산들을 검색한다. 마지막으로, 각 클래스 및 인터페이스 간의 관계들을 검색한다. 클래스 다이어그램 외의 정보 즉, 클래스 다이어그램에 대하여 사용자가 직접 입력한 정보나 클래스나 속성 또는 연산의 이름으로 검색하는 경우는 먼저 소속된 클래스 다이어그램의 식별자를 추출한 후 위의 방법을 적용한다.

4. 구현 및 기존 시스템과의 비교

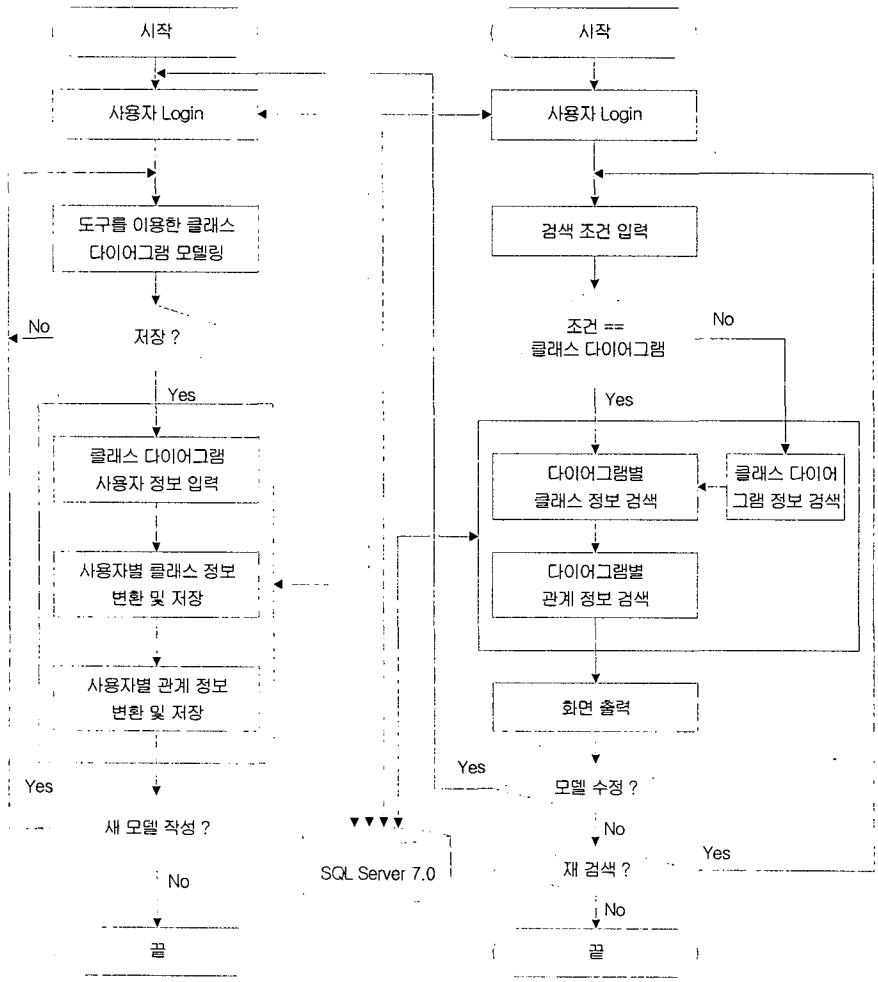
4.1 알고리즘 및 시스템 흐름도

아래의 알고리즘은 관계형 데이터베이스를 이용하여 UML 클래스 다이어그램을 저장

하고 검색하는 과정을 기술한 것이다. 저장을 위한 입력은 클래스 다이어그램을 구성하는 전체 구성 요소가 되고, 결과로서 각각의 테이블에 데이터가 저장된다. 검색에서의 입력은 클래스 다이어그램의 이름을 기준으로 하였고, 출력은 전체 클래스 다이어그램이 된다.

```
begin
{
  $diagram_name = input_diagram_name();          // 클래스 다이어그램의 이름은 사용자 입력
  if (diagram_save == TRUE)                      // 클래스 다이어그램 저장이면
  {
    while (not class_name_set_empty())           // $class_name이 NULL일 때까지
    {
      insert_class_name($class_name, $diagram_name); // 클래스 이름 및 관련된 정보 저장
      while (not attribute_list_empty())         // 현 클래스 내부의 속성 목록(attribute list) 저장
        insert_attribute_name($attribute_name, $class_name);
      while (not operation_list_empty())         // 현 클래스 내부의 연산 목록(operation list) 저장
        insert_operation_name($operation_name, $class_name);
      if (relationship_seek())                   // 현재 다이어그램 내부에 있는 모든 관계 검색, 저장
        insert_relationship($ret_class_name, $class_name, $diagram_name);
    }
  }
  else if (diagram_search == TRUE)               // 클래스 다이어그램 검색이면
  {
    $class_name[] = select_all_class($diagram_name); // 소속된 모든 클래스와 인터페이스 검색
    for each $class_name                         // 각각의 클래스에 대하여
    {
      select_all_attribute($class_name);         // 소속 속성 검색
      select_all_operation($class_name);         // 소속 연산 검색
    }
    select_all_relationship($diagram_name);      // 소속 관계 검색
  }
}
end
```

그림 7은 본 논문에서 제안한 방법을 구현한 시스템의 전체 흐름도(flowchart)이다. UML의 클래스 다이어그램을 관계형 데이터베이스로 저장하기 위한 시스템의 흐름도와 관계형 데이터베이스로부터 클래스 다이어그램 정보를 검색하는 흐름도를 나타내고 있다. 클래스 다이어그램의 보안을 위하여 사용자 로그인 과정이 필요하다.



(a) 클래스 다이어그램 저장 흐름도 (b) 클래스 다이어그램 검색 흐름도

그림 7. 클래스 다이어그램 저장 및 검색 흐름도

4.2 시스템 구현

본 논문에서 제안한 방법을 구현하기 위하여 관계형 데이터베이스로는 Microsoft SQL Server 7.0을 사용하였고, 프로그래밍 언어로는 Visual C++ 6.0을 사용하였다. ODBC(Open Database Connectivity)를 통해서 데이터베이스와 연결하고 트랜잭션(transaction)을 처리하기 때문에 관계형 데이터베이스가 바뀌는 경우는 ODBC의 설정을 변경하여 구현한 시스템을 쉽게 사용할 수 있다.

그림 8은 본 논문에서 구현한 시스템을 사용하여 모델링한 무인 운반 차량의 기본적인 운행을 모니터링하는 예제 클래스 다이어그램이다. 클래스 "Sensor"의 하위 클래스로는 장애물을 탐지하는 클래스 "Hurdle_Sensor"와 운반경로를 탐지하는 클래스 "Line_Sensor"가 있다. 이 두 클래스는 "Controller" 클래스와 연관 관계를 지니며 장애물과 경로를 탐지하여 클래스 "Controller"로 탐지 내용을 보내는 역할을 하게 된다. 그리고 "Controller" 클래스는 각각의 차량의 ID에 따라서 센서 클래스를 제어해야 한다. 클래스 "Controller"는 클래스 "Motor", "Display", "Monitoring"과도 연관 관계를 지니고 있다. "Controller"와 "Monitoring" 클래스와의 관계에서 각각의 다중성(multiplicity)은 "1..*", "1"이다. 즉, 현재 하나 이상의 무인 운반 차량을 감시하는 하나의 시스템이 존재한다는 것을 의미한다. "Controller" 클래스 내부에는 현재 위치 정보, 목적지 위치 정보, 장애물 발견 유무, 현재 속도 등을 나타내는 속성과 센서를 통하여 장애물 탐지, 경로 탐지 그리고 차량을 운영하는 연산들이 필요하다. "Monitoring" 클래스에서는 속성으로 화면에 출력할 차량의 ID와 현재 상태 및 위치를 파악하는 연산들로 구성된다.

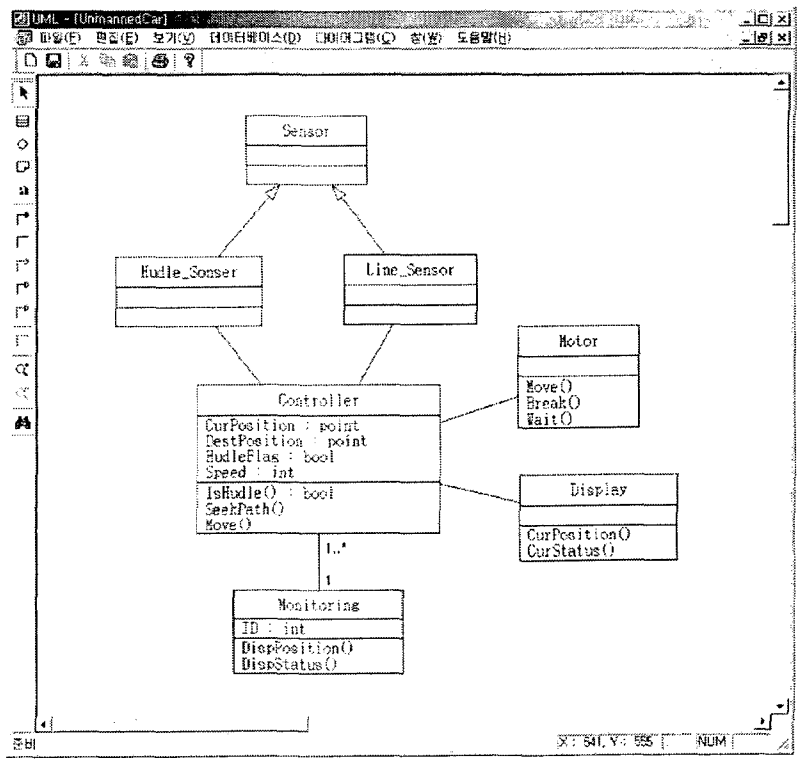


그림 8. 구현한 시스템을 사용한 예제 클래스 다이어그램

4.3 기존 시스템과의 비교

제안한 시스템은 UML을 사용하여 설계된 모델링 정보를 다수의 사용자들이 실시간으로 공유하여 설계에 공동으로 참여할 수 있도록 하기 위해 기존의 파일 저장 시스템이 아닌 관계형 데이터베이스를 사용하도록 하였다. 기존의 파일 저장 시스템을 사용하고 있는 Rose나 Plastic은 UML에서 지원하는 모든 다이어그램을 지원하고 있으며, 순공학과 역공학을 지원하지만 실시간에 모델링 정보를 공유할 수 없다는 단점이 있다. 표 1은 Rose, Plastic 그리고 제안한 시스템을 기능적인 면에서 비교하고 있다.

표 1. 기존 시스템과 제안한 시스템과의 비교

시스템 / 기능	Rose	PLASTIC	제안한 시스템
지원 다이어그램 수	9	8	1
순공학 지원	○	○	X
역공학 지원	○	○	X
데이터베이스 지원	X	X	○
실시간 정보 공유	X	X	○

5. 결론 및 추후 연구과제

UML은 기존의 다양한 시스템 개발 방법론들을 하나의 틀로 통합한 것이다. 특히, 소프트웨어 시스템의 설계, 개발 등을 체계적으로 지원하는 모델링 언어이다. 이러한 UML로 개발된 모델들의 효율적인 사용을 위해서는 모델들을 통합하여 저장하고 관리할 필요성이 있다. 따라서 본 논문에서는 UML 클래스 다이어그램을 저장하고 검색하기 위한 관계형 데이터베이스를 설계하였고, 검색하는 방법을 제안하였다. 제안한 방법에 따라 모델링 정보를 데이터베이스화함으로써 변화하고 있는 컴퓨팅 환경과 개발 방법론에 빠르게 대처할 수 있다. 또한 다수의 사용자가 설계에 참여할 수 있도록 모델링 정보를 공유함으로써 설계 과정의 효율을 향상시키는 데도 기여할 수 있을 것이다. 현재 가장 많은 사용자를 확보하고 있는 관계형 데이터베이스를 사용함으로써 현재의 데이터베이스 시스템에 바로 적용할 수 있다는 장점도 가지게 된다.

본 연구의 추후 과제로는 현재 클래스 다이어그램에 한정된 관계형 데이터베이스로의 저장 및 검색 방법을 UML로 표기된 모든 종류의 다이어그램으로 확대하여 적용하는

것이다. 또한 본 논문에서 제안한 방법에 소프트웨어 순공학(forward engineering) 및 역공학(reverse engineering) 기법을 도입하여 모델링이 곧 프로그래밍과 연결될 수 있도록 하는 것이다. 또한 분산 환경에서의 소프트웨어 개발을 지원할 수 있도록 해야 하며, UML로 생성된 모델들의 OODB(Object Oriented Database)로의 저장 방법도 개발할 필요성이 있다.

참 고 문 헌

[1] J. Rumbaugh, *Object-Oriented Modeling and Design*, Prentice-Hall, 1991.

[2] G. Booch, "Object-Oriented Development", *IEEE Transactions on Software Engineering*, SE-12, pp. 211-221, 1986.

[3] G. Booch, *Object-Oriented Analysis and Design with Application*, 2nd ed., Benjamin Cummings Pub., 1994.

[4] I. Jacobson, *Object-Oriented Software Engineering : A Use Case Driven Approach*, Addison-Wesley, 1992.

[5] http://www.omg.org - OMG Home Page, specification for UML and related modeling standards, such as MOF and XMI.

[6] http://www.rational.com/uml - Rational Software's UML Resource Page.

[7] http://www.plasticsoftware.com - Plastic Software's Home Page.

[8] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 3th ed., McGraw-Hill, 1996.

[9] KMK 정보산업연구원, *관계형 DB의 객체지향기법*, 삼각형, 1995. 5.

[10] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 1997.

[11] M. Priestley, *Practical Object-Oriented Design with UML*, McGraw-Hill, 2000.

[12] http://www.uml.co.kr - UML 및 객체지향 관련 학술, 토론 사이트.

[13] 왕창종, *소프트웨어 공학*, 정익사, 1998.

[14] 강설문, 김태희, "객체지향 소프트웨어 개발 방법론의 표준화 : UML (Unified Modeling Language)", *정보처리논문지*, 제5권 제5호, pp. 64-73 1998. 8.

[15] 이성대, 박휴찬, "UML 모델의 저장 및 질의를 위한 관계형 데이터베이스 설계 및 구현", *2000 추계학술발표 논문집*, 상권, pp. 79-82, 한국정보처리학회, 2000. 10.