2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

Windows 버전 OODesigner의 설계 및 구현

김기욱*, 김태균*
*부산외국어대학교 컴퓨터공학과
e-mail:ktg@taejo.pufs.ac.kr

Design and Implementation of Windows Version OODesigner

Ki-Ook Kim*, Tae-Gyun Kim*
*Dept of Computer Engineering, Pusan University of Foreign Studies

요약

본 논문에서는 기존에 구현한 유닉스 버전 CASE(Computer Aided Software Engineering) 도구인 OODesigner를 윈도 버전으로 이식한 연구 결과에 대하여 논한다. 유닉스 버전 OODesigner는 Rumbaugh가 제안한 OMT(Object Modeling Technique) 기법을 지원하기 위해 구현되었지만 윈도 버전은 UML(Unified Modeling Language)을 지원할 수 있도록 기능이 향상되었다. 본 논문은 구현된 윈도 버전 OODesigner의 설계 문서와 구현 결과를 제시하는 것을 목적으로 한다.

1. 서론

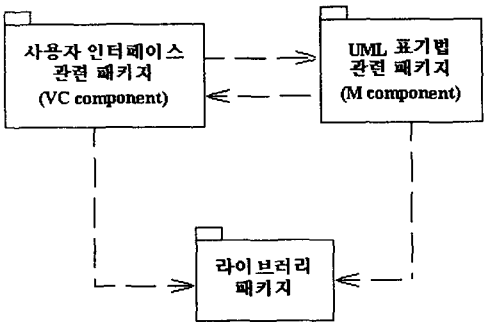
객체 지향 패러다임은 많은 응용 분야에 걸쳐 소프트웨어 엔지니어로 하여금 소프트웨어를 생산성 있게 개발할 수 있도록 하는 환경을 제공해 준다. 과거에 이루어진 연구 결과로 볼 때 객체 지향 기법은 구조적 기법에 비해서 문제를 풀기 위한 좀 더 자연스러운 방법을 마련해 줌을 알 수 있다. 시스템 개발자는 객체 지향 기법을 통하여 추상화, 정보 은닉, 모듈화, 계층 구조화, 지속성, 동적 바인딩, 다형 개념과 같은 원리를 종합적으로 적용할 수 있다. 객체 지향 기법을 통하여 소프트웨어를 개발하면 시스템의 보수유지가 용이해 질뿐더러 소프트웨어의 재사용을 통한 생산성 향상을 이룰 수 있다. 그러나 성공적인 객체 지향 프로젝트의 수행은 객체 지향 언어와 객체 지향 방법론 그리고 객체 지향 CASE 도구를 종합적으로 적용하였을 때 이루어질 수 있다. 즉 단순히 객체 지향 언어로 프로그래밍을 했다고 해서 그 소프트웨어가 객체 지향적이라고 할 수 없으며 오히려 경우에 따라서는 객체 지향 언어로 작성된 소프트웨어가 구조적 언어로 작성된 소프트웨어보다 보수유지하기가 더 어려운 경우도 있다. 그러한 경우에 프로젝트가 실패하는 이유는 시스템의 분석과 설계 시에 시스템을 객체 지향적으로 명세하지 않고 객체 지향 코딩을 수행하기 때문이다.

객체 지향적인 시스템 개발의 본질은 대규모의 시스템을 구성하는 부속 객체들을 인식하여 이를 객체로 표현하고 그들 간의 상호 관계 및 임무가 적절한 복잡도 내에서 전체 시스템 내에 고르게 분산될 수 있도록 배치하는 것이다. 이러한 과정은 객체 지향 방법론이나 객체 지향 모델링 언어를 이용하여 이루어질 수 있다. 객체 지향 기법의 선각자들은 1980년대 중반 이후로 1990년대 중반까지 약 20 여 개의 객체 지향 방법론을 각자 고안하여 발표한 바 있다. 1990년대 중반에 가장 인기가 있었던 객체 지향 방법론 중의 하나는 Rumbaugh의 OMT 기법 이었다. 본 연구의 이전에 1993년부터 OMT 기법을 지원하는 CASE 도구를 구현하고자 하는 목적으로 OODesigner라는 도구를 유닉스 환경에서 C++ 언어와 OSF/Motif 툴킷을 이용하여 구현하여 public domain에 공개한 바가 있다. OODesigner는 OMT의 표기법 작성 기능외에도 C++ 언어의 역공학 기능 및 코드 생성 기능, 정보 저장소 기능, 그리고 C++ 메트릭스(metrics) 수집 기능을 갖고 있다. 이 도구는 1995년 이후에 refactoring 과정을 거쳐 재구조화

되었기 때문에 좀더 바람직한 시스템 구조를 갖게되었다. 재구조화된 유닉스 버전 OODesigner는 기기 독립성이 유지되도록 시스템이 만들어졌기 때문에 윈도 버전으로 이식하는 것이 가능하다고 판단되어 본 연구를 수행하게 되었다. 이러한 점에서 본 연구는 Unix 버전 CASE 도구의 시스템 이식을 통한 UML[1] 도구의 개발에 관련된 것이다. 윈도 버전에서 구현된 내용은 UML 표기법의 Class Diagram, Use Case Diagram, Sequence Diagram, Collaboration Diagram, State Diagram, Activity Diagram, Deployment Diagram 작성 기능과 C++/Java 코드의 자동 생성 기능이다. 본 논문의 2장에서는 시스템의설계에 대하여 논하며 3장에서는 구현 결과를 제시하고 4장에서는 결론과 함께 차후 연구 방향에 대하여 기술한다.

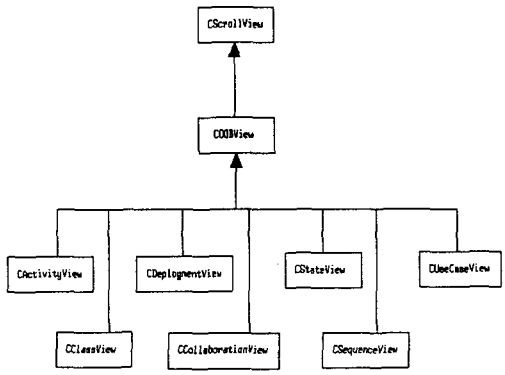
2. 시스템 설계

본 연구에서 구현하고자 하는 소프트웨어는 MVC(Model-View-Controller) 의 구조를 갖는다. Model 요소는 시스템의 응용 목적을 위해 정의된 클래스들을 의미하고 View 요소는 그 모델을 화면에 보여주는 역할을 수행하는 기능을 의미하며 Controller 요소는 소프트웨어의 작동 시에 사용자에 의해 발생되는 이벤트들을 처리하는 기능을 의미한다. 특히 View 와 Controller 부분의 기능은 주로 사용자 인터페이스와 관련된 부분이기 때문에 View 와 Controller 구성 요소는 함께 취급될 수 있다. 본 도구의 개략 설계는 MVC 모델을 기반으로 이루어졌다. (그림 1)은 OODesigner 구현을 위한 개략적인 구조이다.



(그림 1) 윈도 버전 OODesigner의 개략적인 구조

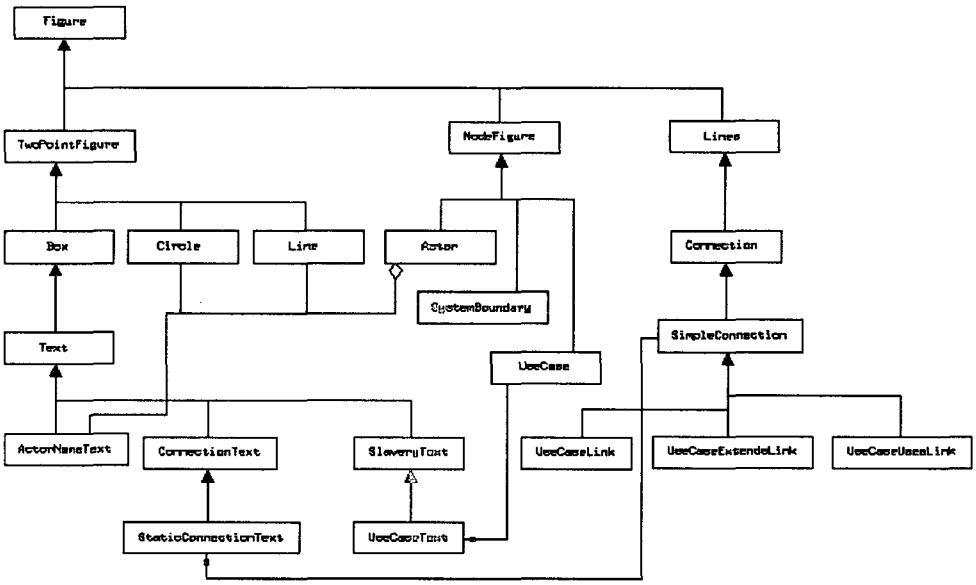
본 연구에서 구현된 OODesigner는 다양한 표기법을 작성하도록 지원하는 여러 종류의 화면을 필요로 하기 때문에 Visual C++의 MDI(Multiple Document Interface) 방식으로 설계되었다. 본 도구에서는 UML의 표기법 종류 수 만큼의 사용자 인터페이스 컴포넌트를 가져야 한다. 예를들어 클래스 도표를 작성하기 위한 화면의 속성은 상태 도표를 작성하기 위한 화면의 속성과 다르기 때문에 서로 다른 프레임(frame)과 서로 다른 뷰(View)를 가져야 한다. 그런데 클래스 도표를 위한 뷰의 기능과 상태 도표를 위한 뷰의 기능은 완전히 다르지도 않으며 공통적으로 수행해야 하는 부분도 있기 때문에 이러한 상황은 상속을 이용하여 설계될 수 있다. 본 도구 설계의 큰 특징 중의 하나는 이러한 점을 고려하여 UML 표기법에 따른 다수의 뷰와 다수의 프레임을 상속을 이용하여 효율적으로 조직하였다는 점이다. 이와같이 설계된 구조로 인해 본 도구의 구현 임무가 적절히 분담이 될 수 있었다. (그림 2)는 OODesigner에서 사용되는 뷰들의 상속 관계이다.



(그림 2) 뷰들의 상속 관계

OODesigner는 UML에서 정의된 모든 표기법을 지원하며 이러한 표기법의 기능을 제공하도록 설계된 클래스의 집합체가 표기법 관련 패키지이다. 이 패키지에 속하는 클래스들은 모두 Figure 클래스로부터 상속되는 클래스들로 약 80개의 클래스가 있다. 이들 클래스들은 UML에서 정의된 도표 별로 Use Case Diagram을 위한 모델, Class Diagarm을 위한 모델, Sequence Diagram을 위한 모델, Collaboration Diagarm을 위한 모델, State Diagarm을 위한 모델, Activity Diagarm을 위한 모델, Deployment Diagarm을 위한 모델로 나뉘어 설계되었다. (그림 3)은 이중에서 Use Case Diagram을 위한 설계 결과이며 이들 클래스의 역할에 대한 설명은 다음과 같다.

- Figure : 이 클래스는 표기법에서 공통적으로 사용되는 인터페이스를 제공하며 모든 표기법에 공통으로 소속된 데이터 멤버들을 정의한다.



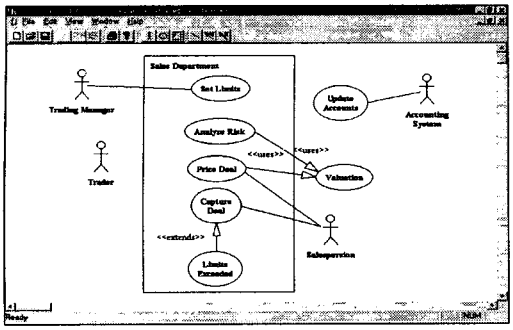
(그림 3) Use Case Diagram 표기법 설계를 위한 클래스 도표

- TwoPointFigure : 이 클래스는 이차원적인 좌표값을 갖는 객체를 위한 자료구조와 멤버 함수를 정의한다. 이 클래스의 중요한 데이터 멤버는 이차원 도형의 꼭지점 좌표이다.
- Box, Circle, Line : 이들은 각각 사각형, 원, 라인 객체들을 나타내기 위한 것이다. 이들 중에서 특히 사각형 클래스는 다른 복잡한 표기법들의 툴을 제공하는데 재사용된다.
- NodeFigure : UML 표기법에 정의되는 모든 표기법은 크게 두 부류에 속하게된다. 그 하나는 도표상에서 노드의 역할을 하는 것이고 다른 하나는 연결선의 역할을 하는 것이다. 이 클래스는 노드 역할을 하는 모든 객체의 공통 인터페이스를 제공한다.
- Actor, SystemBoundary, UseCase : 이들 세 객체는 Use Case Diagram에서 사용되는 노드 객체들이다. 이들은 클래스 명칭에서 알 수 있는바와 같이 각각 사람 모양의 Actor 와 사각형의 틀로 표현되는 시스템 그리고 타원으로 표시되는 use case 객체를 표현하기 위한 것이다.
- Text, ActorNameText, ConnectionText, StaticConnectionText, SlaveryText, UseCaseText : 이들은 화면 상에 문자열을 입력하거나 그려주기 위한 텍스트 클래스들이다. 이들 중에서 Text 클래스는 화면의 문자열 처리를 위한 모든 데이터 멤버와 공통의 함수를 포함하며 나머지 클래스들은 특정한 표기법 객체의 부속으로 소속되어 표기법의 문맥에 맞는 서비스를 제공하는 텍스트 객체들을 표현하기 위한 것이다.
- Lines : 이 클래스는 여러개의 선분으로 이루어지는 선을 나타내기 위한 클래스이다. UML 표기법에서 사용되는 각종 연결선들은 이 클래스로부터 상속된다.
- Connection : 이 클래스는 표기법에서 논리적으로 노드들을 연결하는 의미를 갖는 연결선을 표현하기 위한 클래스이다.
- SimpleConnection : 이 클래스는 연결선이 한 개의 선분으로만 표현되는 경우를 처리하기 위한 클래스이다.

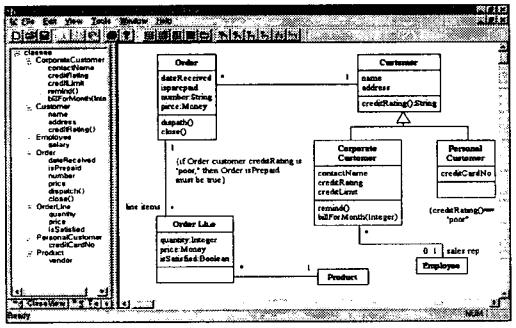
3. 구현 결과

본 장에서는 연구 수행 결과를 구현된 도구의 실행 예를 중심으로 기술한다. 본 연구는 소프트웨어의 구현에 관한 연구이기 때문에 도구의 실행 화면을 통하여 연구 결과를 명확히 제시할 수 있을 것으로 판단된다. 본 절에서 제시된 화면들은 Martin Fowler의 "UML Distilled: Applying the Standard Object Modeling Language" 책에 있는 모델들을 OODesigner를 이용하여 작성한 것들이다.

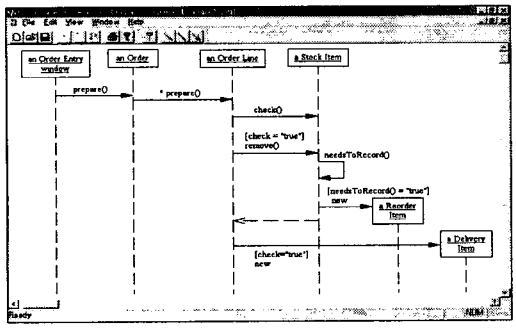
OODesigner의 작동을 위한 사용자 인터페이스는 기존에 존재하는 상업적인 CASE 도구와 유사하게 구현되었다. 논문의 분량에 대한 제약 때문에 화면들에 대한 설명은 생략한다.



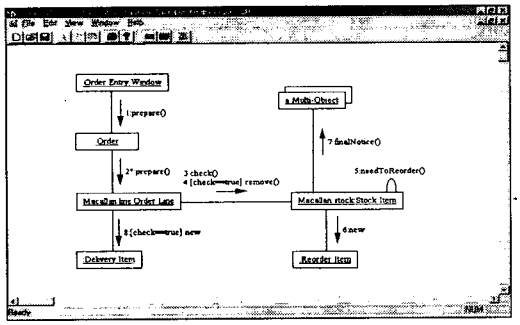
(그림 4) Use Case 도표 작성기



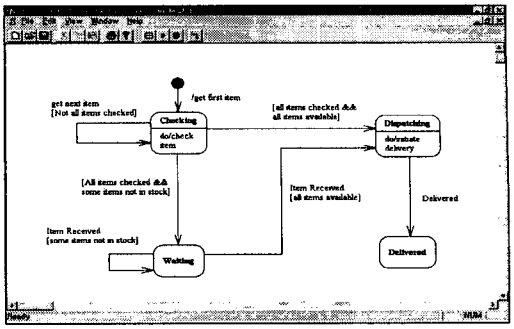
(그림 5) 클래스 도표 작성기



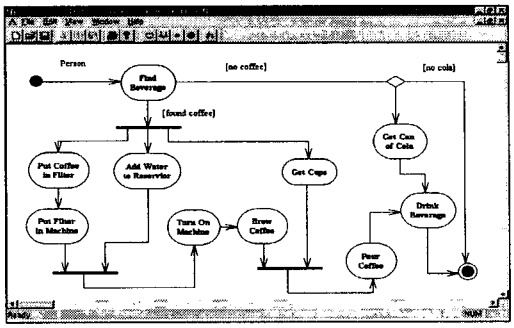
(그림 6) Sequence 도표 작성기



(그림 7) Collaboration 도표 작성기



(그림 8) 상태 도표 작성기



(그림 9) 행위 도표 작성기

4. 결론

본 논문에서는 UML을 지원하는 CASE 도구를 윈도 플랫폼에서 구현한 결과를 제시하였다. 본 연구의 특징은 특히 이전 연구 결과물인 유닉스 버전 OODesigner를 이식하는 작업을 통하여 PC 버전을 구현하였다는 점이다. 본 연구에서 목표로 설정한 내용은 다음과 같으며 이들 목표는 본 연구를 통하여 성공적으로 달성되었다.

- Unix 버전의 기존 기능을 윈도 버전으로 이식
- UML 표기법 작성 기능 구현
- C++/Java 에 대한 코드 생성 기능 구현

현재 구현 결과물에 존재하는 문제점은 시스템의 안정성에 관한 것이다. 추후의 계속적인 연구를 통하여 시스템의 신뢰성이 확보될 수 있도록 보완할 계획이며 본 도구를 메타 CASE 도구로 변환하기 위한 연구도 지속적으로 추진할 계획이다.

참고문헌

[1] M. Fowler "UML Distilled: Applying the Standard Object Modeling Language" Addison-Wesley, 1997