OLE 컨테이너 기능을 지원하는 CASE 도구의 설계 및 구현

강지윤°, 김태균
부산외국어대학교 컴퓨터공학과
e-mail: {kangjy°,ktg}@saejong.pufs.ac.kr

Design and Implementation of a CASE Tool with OLE Container Functionality

Jeeyoon Kang°, Taegyun Kim
Dept of Computer Engineering, Pusan University of Foreign Studies

요 약

본 논문에서는 기존에 구현된 CASE(Computer Aided Software Engineering) 도구인 OODesigner에 OLE/COM 기술을 이용하여 복합문서 지원 기능을 추가로 구현한 연구 결과에 대하여 논한다. OLE 컨테이너 기능을 구현함으로써 OODesigner는 다른 소프트웨어에 의해 만들어진 문서 객체를 포함하여 복합문서를 작성할 수 있다. 본 연구의 결과로 구현된 복합문서 지원 기능을 갖는 OODesigner는 윈도 시스템에서 실행되는 각종 응용 프로세스와 상호 협동 작용을 하며 수행될 수 있기 때문에 기존의 CASE 도구가 지원하지 못하는 유용한 문서화 기능을 제공할 수 있다. 즉 객체 지향 관련 설계 문서의 내용을 OODesigner로 편집함과 동시에 OLE 기능을 지원하는 각종 서버 소프트웨어를 연결하여 수행함으로써 일관성 있고 편리한 문서화 작업을 수행할 수 있다.

1. 서론

1980년대 중반부터 객체지향 언어의 사용이 확산되고 많은 객체지향방법론들이 발표된 이후로 객체지향 기술이 소프트웨어 공학자들의 가장 큰 관심분야의 하나로 떠오르게 되었다. 1997년 객체 지향 방법론들에 대한 통합 노력의 결과로 정의된 Rational 사의 UML은 OMG에 의해 객체 지향 설계를 위한 표준화된 기법으로 선정된 이후에 다양한 분야에서 사용되고 있다.[1]

UML을 비롯한 객체 지향 방법론들은 그 방법론을 지원하는 유용한 CASE 도구의 지원이 있을 때 효용이 극대화되기 때문에 많은 업체가 상업적인 CASE 도구들을 개발하여 판매하고 있으며 Rational 사의 Rational Rose나 Together Soft 사의 Together 같은 도구가 현재 널리 사용 중이다.

본 연구의 이전에 본 연구자에 의해 수년 동안 이루어지고 있는 OODesigner의 구현에 관한 연구도 객체 지향 방법론을 지원하는 CASE 도구의 구현을 위한 연구로서 현재까지 나름대로의 성과를 얻은바 있다.

모든 CASE 도구들이 당면하고 있는 문제점은 도구 자체가 현재 보편화되고 있는 기술인 COM/OLE[2,3] 기능을 지원하지 못하고 있다는 점이다. 현재 사용되고 있는 모든 도구는 다른 응용 소프트웨어와의 연결을 목적으로 개발되어있지 않기 때문에 도구에서 필요한 모든 기능을 도구 내에서 구현해야하는 부담을 갖고 있다.

예를 들어 UML로 작성된 설계 문서에 설계자의 음성을 이용하는 문서화를 하고자하는 경우 컴포넌트 기술을 적용한 CASE 도구가 존재하게 되면 MS사의 녹음기 응용 프로그램을 연결해서 사용하면 되지만 컴포넌트 기술을 적용하지 않은 도구는 소리를 저장하고 재생하는 기능을 직접 구현하여 CASE 도구의 부속 기능으로 추가하여야 한다.

이러한 예에서와 같이 멀티미디어 데이터, 차트, 스프레드시트와 같은 자료를 포괄적으로 처리할 수 있는 다양한 문서화가 간단한 소프트웨어 설계 환경의 구축은 CASE 도구에 COM/OLE 기능을 추가함으로써 가능하다. COM/OLE 기술을 포함한 컴포넌트 관련 연구가 성숙되어 있음에도 불구하고 CASE 도구에 이 기술을 접합하고자하는 시도는 미미한 실정이며 현재 CASE 시장에 나와 있는 모든 CASE 도구 중에 복합문서 지원 기능을 제공하는 도구가 없다는 점에서 이러한 기능을 제공하는 도구 개발의 필요성이 있다.

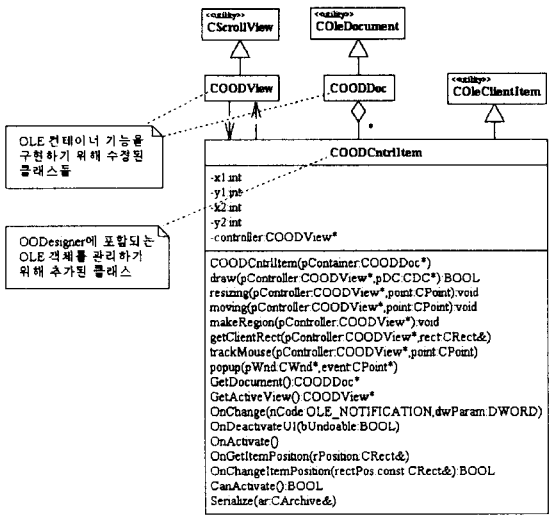
본 논문의 2장에서는 시스템의 설계에 대하여 논하며 3장에서는 구현 결과를 제시하고 4장에서는 결론과 함께 차후 연구 방향에 대하여 기술한다.

2. 시스템 설계

본 연구는 한국과학재단 목적기초연구(R05-2002-000-00497-0) 지원으로 수행되었음.

기존의 OODesigner는 Visual C++로 구현되었으며 약 200개의 클래스들로 구성된다. OODesigner는 MVC(Model View Controller) 패러다임에 따른 구조 설계에 따라 클래스간의 결합도가 느슨하게 설계되었기 때문에 시스템의 유지보수가 용이하다. MVC 구조에 따라 도구 사용 시에 사용자는 컨트롤러에 속하는 객체들을 이용하여 모델 정보를 입력하고 그 정보는 도큐먼트에 저장되며 저장된 내용이 뷰에 디스플레이 되는 방식으로 도구가 작동된다. 이러한 설계 내용 중에서 OLE 기능의 추가 시에 컨트롤러와 뷰는 크게 영향을 받지 않았으나 도큐먼트 부분에서는 많은 수정이 이루어졌다.

OODesigner에 OLE 컨테이너 기능을 추가하는 것은 예상되는 노력에 비해 상대적으로 용이하게 이루어졌다. OLE 컨테이너 기능의 추가를 위하여 COODApp 메인 클래스에 OLE 라이브러리를 초기화시키기 위한 코드가 추가되었으며 CMainFrame 클래스에 OLE 객체를 위한 메뉴 구현 코드가 추가되었다. OLE 컨테이너 기능 추가를 위해 핵심적으로 변경된 설계 사항은 (그림 1)과 같다. (그림 1)에서 보는 바와 같이 기존에 구현되어 있던 COODView클래스와 COODDoc 클래스에 OLE 아이템을 처리하기 위한 함수들이 추가로 필요하여 기능을 수정하게 되었으며 COleClientItem클래스로부터 상속받는 COODCntrlItem 클래스를 새로이 구현하여야 했다.



(그림 1) 컨테이너 기능을 위한 클래스 다이어그램

(그림 1)에서 도큐먼트와 관련된 수정된 사항은 다음과 같다. 우선 COODDoc 클래스의 상위 클래스가 CDocument로부터 COleDocument 클래스로 바뀌었는데 그 이유는 OLE 기능의 구현 시에 구조화 된 저장소(storage)를 사용하기 때문이다. COODDoc 클래스를 수정하면서 기존에 구현되었던 함수인 Serialize(), OnOpenDocument(), OnNewDocument() 와 같은 함수들이 OLE 객체들의 리스트를 관리하며 구조화된 저장소를 처리할 수 있도록 변경되었으며 setRootStg(), OnNewEmbedding(), OnOpenEmbedding() 과 같은 몇몇 함수가 루트 저장소를 생성하여 초기화시키기 위해 추가로 구현되었다.

(그림 1)에서 뷰와 관련된 수정 사항은 다음과 같다. OODView 클래스에는 주로 OLE 객체를 생성하거나 객체의 상태 변화를 일으킬 수 있는 사용자 입력을 처리하는 기능들이 추가되었다. 즉 OnInsertObject() 함수를 통하여 OLE 객체를 삽입하고, OnObjectEdit(), OnObjectOpen() 함수들을 통하여 삽입된 OLE 객체가 활성화 될 수 있게 하였으며, 사용자의 마우스 입력을 통해 OLE 객체들의 크기나 상태를 변경할 수 있도록 기존의 이벤트 처리 함수들이 변경되었다.

(그림 1)에서 새로 추가된 COODCntrlItem 클래스는 OODesigner에 삽입되거나 연결되는 OLE 객체를 관리하기 위한 목적을 갖는다. 이 클래스가 수행하는 기능은 삽입된 OLE 객체의 위치와 크기를 관리하고 필요한 경우 디바이스 컨텍스트에 그 객체를 그리거나 객체를 활성화하는 것이다. (그림 1)의 COODCntrlItem 내용은 전체 구현 내역 중에서 핵심적인 멤버들만을 제시한 것이다.

■ x1,y1,x2,y2: 이들 데이터 멤버는 OODesigner에 삽입된 OLE 객체의 위치와 크기를 저장하기 위한 것이다.

■ viewPtr: 이 데이터 멤버는 삽입된 OLE 객체가 포함된 뷰의 위치를 가리키는 포인터이다. OODesigner는 MDI(Multiple Document Interface) 방식으로 구현되어 있기 때문에 실행시에 여러개의 도큐먼트와 뷰가 이용된다. viewPtr은 OLE 객체가 자기 자신을 관리하는 뷰와 도큐먼트를 찾기 위해 사용되는 포인터이다.

■ draw(): 이 함수는 COleClientItem::draw() 함수로 작업을 위임함으로써 삽입된 OLE 객체를 해당 뷰에 그려준다.

■ resizing(): 이 함수는 삽입된 OLE 객체의 크기를 조정하는데 사용된다.

■ moving(): 이 함수는 삽입된 OLE 객체를 움직이는데 사용된다.

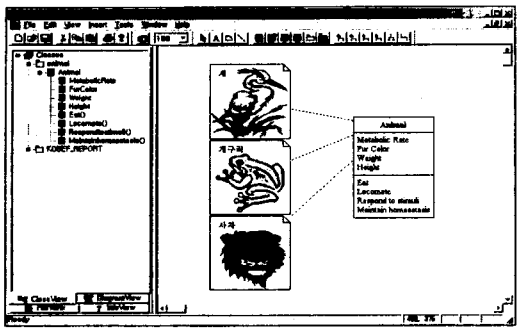
■ makeRegion(): 이 함수는 삽입된 OLE 객체의 인식을 위해 사용되는 영역을 구한다.

이상과 같은 내용은 복합 문서 지원 기능을 추가하는 과정에서 이루어진 개략적인 설계 결과이다. 다음장에서는 구현 결과들을 제시한다.

3. 구현 결과

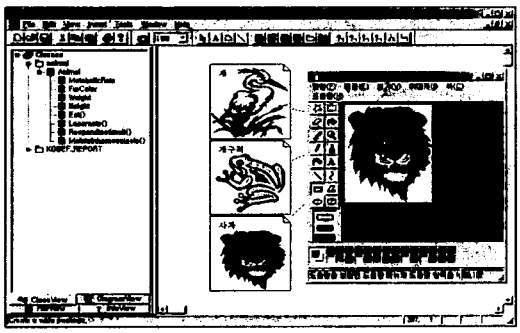
현재까지의 연구수행 결과 OLE 컨테이너 기능의 구현을 완벽히 이룰 수 있었다. OODesigner에 OLE 컨테이너 기능을 구현함으로써 OODesigner는 오디오, 비디오 기능을 이용한 도큐멘테이션이 가능해졌으며 아울러 윈도에서 제공되는 그림판, 엑셀, 그래픽 편집기 등의 객체를 UML 문서에 연결할 수 있게되었다. 따라서 OODesigner는 음성, 동영상, 그림, 워드 문서를 편집할 수 있는 소프트웨어와 함께 실행되면서 소프트웨어

에 개발자에게 막강한 문서화 환경을 제공할 수 있다. 본 절에서는 그 동안의 연구 결과를 사용 예와 실행화면 중심으로 제시한다.



(그림 2) OODesigner에 그림 객체가 삽입된 예

OLE 컨테이너 기능을 통해 제공된 기능 중의 하나는 (그림 2)에서와 같이 UML 문서에 그림 파일을 삽입하는 것이다. (그림 2)는 Animal 클래스를 설계 시에 Animal 객체의 예가 어떠한 것이 있는가를 알기 쉽게 보여주고 있다.

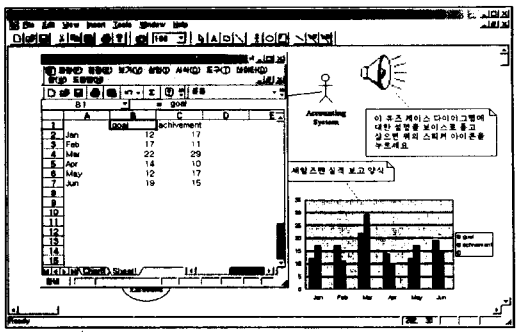


(그림 3) 그림 객체의 오픈 방식 활성화 예

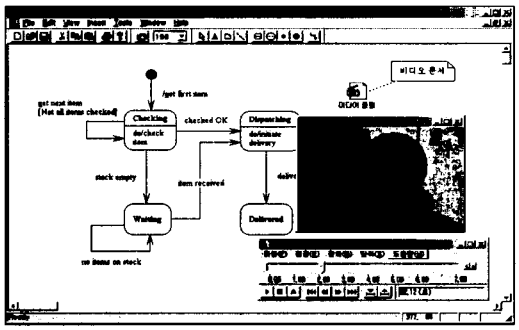
(그림 3)의 경우는 오픈 방식에 의한 OLE 객체 활성화 상황을 보여준다. 오픈 방식으로 객체를 활성화하기 위해서는 삽입된 OLE 객체가 제공하는 <open> 메뉴를 이용한다. (그림 3)의 경우에 호랑이 그림은 원본 그림 파일과 연결되어 있는 경우의 예이며 화면에서와 같이 호랑이의 눈 부분을 그림판을 통해 수정하면 OODesigner의 뷰 상에서 수정이 이루어짐과 동시에 원본 파일이 변경됨을 확인 할 수 있다.

OLE 컨테이너 기능을 통해 제공되는 다른 기능은 녹음기나 엑셀과 같은 도구를 이용하여 작성된 객체를 삽입하는 것이다. (그림 4)의 화면은 엑셀로 작성된 그래프 객체를 활성화하는 상황을 보여준다. 화면에서와 같이 OODesigner가 기존에 Window에서 제공되는 도구들과 연동되어 실행될 수 있기 때문에 문서 작성의 편의성과 일관성을 달성할 수 있다.

마지막으로 제시된 (그림 5)의 화면은 캠코더로 작성된 비디오 도큐먼트의 활성화 상태를 보여준다.



(그림 4) OODesigner에 포함된 엑셀 객체의 활성화



(그림 5) 비디오 문서 활성화의 예

본 장에서는 그 동안의 구현 결과를 실행 화면 위주로 제시하였다. 본 절에서 제시된 내용으로 볼 때 본 연구의 목표인 OLE 컨테이너 기능의 개발이 성공적으로 이루어졌음을 알 수 있다.

4. 결론

본 논문에서는 기존에 구현된 UML CASE 도구인 OODesigner에 OLE 컨테이너 기능을 추가로 구현한 연구 결과에 대하여 논하였다. OLE 기능의 구현을 통하여 좀더 유용한 문서화 환경이 제공됨으로써 소프트웨어의 생산성 향상에 이바지할 것으로 기대된다. 본 연구의 차후 목표는 OLE 서버 기능을 추가로 구현하는 것이다.

참고문헌

[1] Martin Fowler, *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley, 1997

[2] Richard Grimes, et al., *Beginning ATL COM Programming*, Wrox, 1999.

[3] D. Chappell, *Understanding ActiveX and OLE*, Microsoft Press, 1997.