韓國 컴퓨터情報學會 論文誌
第12卷 第6號, 2007. 12.

2007-12-6-1-5

복합 문서 지원 기능을 갖는 CASE 도구의 설계 및 구현

조 장 우*, 김 태 균**

Design and Implementation of a CASE Tool with Compound Document Support

Jang-wu Jo*, Taegyun Kim**

요 약

객체지향 기술의 도래 이후로 많은 연구 개발 과제들을 통해 객체지향 기술이 생산성과 재사용성 향상에 크게 도움이 되고 있음이 판명되고 있다. 객체지향 기술과 관련된 다양한 연구 분야가 있으며 그 중에서 두 가지 중요한 분야가 CASE 도구에 관련된 분야와 컴포넌트 기반 기술이다. 본 논문은 이 두 기술을 결합하기 위하여 CASE 도구에 컴포넌트 기술을 적용한 시도에 대해 논한다. 본 논문에서는 기존에 개발되었던 객체 지향 CASE 도구인 OODesigner에 COM/OLE 기반의 복합문서 지원 기술을 지원하도록 추가로 개발한 연구 결과를 제시한다. OLE 컨테이너와 서버 기능을 제공하는 OODesigner는 윈도우 시스템의 응용 프로그램들과 상호 협동 작용을 하며 수행될 수 있기 때문에 다른 CASE 도구들보다 막강한 문서화 기능을 제공할 수 있다. OODesigner는 단지 UML 다이어그램의 설계를 위한 용도로 사용될 뿐 아니라 녹음기나 엑셀과 같이 OLE 기능이 제공되는 다양한 소프트웨어와 협력함으로써 좀 더 표현력이 뛰어난 문서화를 가능하게 한다. 그 결과 이 도구를 통해 강력하고 일관성 있는 문서화 작업이 이루어질 수 있다.

Abstract

Since the advent of object-oriented(OO) technology, research and development projects have turned out that OO technology could importantly contribute in productivity and reusability improvement. There are various research areas related to OO technology. And two of major research fields in this areas are concerned in issues for CASE tools and component-based technologies. This paper discusses a trial of applying component-based technology to a CASE tool. This paper proposes the design and implementation issues obtained while we have incrementally developed OODesigner, an OO CASE tool, with compound document support functionality based on COM/OLE technology. As OODesigner with OLE container/server functionality has interoperability with other application programs of

• 제1저자 : 조장우
• 접수일 : 2007.10.25, 심사일 : 2007.11.16, 심사완료일 : 2007.11.21.
* 동아대학교 컴퓨터공학과 부교수, ** 부산외국어대학교 컴퓨터공학과 교수

Windows system, it can provide more powerful documentation environment than other CASE tools. OODesigner can be used not only to design UML diagrams, but also enables us to make more expressive documentation cooperatively with various kinds of OLE-enabled software like Recorder and Excel. Therefore powerful and consistent documentation activity can be achieved with the tool.

▸ Keyword : CASE 도구(CASE Tool), UML, COM/OLE, 복합문서지원(Compound Document Support)

Ⅰ. 서 론

객체지향 기술이 소프트웨어 개발의 주된 패러다임으로 자리 잡으면서 다양한 객체 지향 방법론들과 이를 지원하는 CASE 도구들이 개발되었다. 1980년대 후반에 발표된 대표적인 객체지향 방법론들로는 Rumbaugh의 OMT(Object Modeling Technique)[1], Coad의 OOA(Object Oriented Analysis)[2], Wirfs-Brock의 RDD(Responsibility Driven Method)[3] 등이 있고, 이들 방법론들에 대한 통합 노력의 결과로 UML(Unified Modeling Language)[4]이 정의되었다. 특히 UML은 OMG(Object Management Group)에 의해 객체 지향 설계를 위한 표준화된 기법으로 선정되었다. 그리고 생산성을 향상시키기 위하여 UML을 비롯한 객체 지향 방법론들을 지원하는 CASE(Computer Aided Software Engineering) 도구들이 개발되었으며, IBM의 Rational Rose[5]나 볼랜드 사의 Together[6] 같은 도구가 현재 널리 사용 중이다. 본 논문에 의해 연구 개발되고 있는 OODesigner도 객체 지향 방법론을 지원하는 CASE 도구의 하나로써 1990년대 중반에 배포된 이후에 국내외에서 몇몇 사용자를 확보하고 있다[7, 8].

기존의 CASE 도구들이 컴포넌트의 설계와 제작을 위한 기능의 제공하기 위해 노력하고 있음에도 불구하고 CASE 도구 자체가 컴포넌트로 구현된 경우는 없다. 즉 기존의 도구들이 컴포넌트로서 사용되도록 개발되지 않았기 때문에 복합 문서 지원(Compound Document Support) 기능과 같은 유용한 기술을 제공하지 못하고 있다. 컴포넌트 기술은 현재 보편화되고 있는 기술로서, 다른 응용 소프트웨어와의 상호 운용성(interoperability)을 목적으로 한다. 그러나 기존의 CASE 도구들은 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 복합 문서를 지원하는 CASE 도구는 기존에 존재하는 녹음기 응용 프로그램을 연결해서 사용하면 되지만 복합 문서를 지원하지 않은 CASE 도구는 소리를 저장하고 재생하는 기능을 직접 구현하여 CASE 도구의 부속 기능으로 추가하여야 한다.

한편으로 1970년대 발표되었던 PSL/PSA(Problem Statement Language/Problem Statement Analyzer)[9], RSL/REVS(Requirement Statement Language/Requirement Engineering Validation System)[10]과 같은 CASE 도구의 원조들이 추구했던 자동화 도구의 목적이 목적 시스템을 위한 추상화된 문서화 작업을 지원하기 위함이었음은 분명한 사실이다. 30 여 년 전에 이루어졌던 텍스트 위주의 문서화와 방법론에서 정의된 표기법의 도형만으로 분석 및 설계 문서 작성을 답습하고 있는 대부분 CASE 도구의 상황은 멀티미디어의 사용이 대중화 되어 있는 현재 기술로 볼 때 개선의 필요성이 시급하다. 즉 목적 시스템 구축을 위한 UML 설계 문서의 작성 기능 외에 추가로 이미지, 음성, 동영상을 이용하여 설계된 문서를 편리하고 자연스럽게 설명할 수 있는 기능이 CASE 도구에 추가되면 CASE 도구의 유용성이 배가될 것이다. 다행스럽게도 이미지, 음성, 동영상을 편집하고 관리하는 응용 소프트웨어들은 이미 MS 윈도시스템에서 다수 제공되고 있다. 따라서 이들 응용 소프트웨어들을 OLE 서버 객체로 인식하여 OODesigner의 도큐먼트에 포함되도록 컨테이너 기능을 구현하는 것이 본 논문의 한 가지 목표이다. 또한 OODesigner 자체를 컴포넌트 화하여 훈글이나 MS 워드와 같이 OLE 컨테이너로 작동하는 문서에 OODesigner의 설계 문서가 삽입되거나 링크되게 할 수 있도록 서버 기능을 구현하는 것이 또 하나의 목표이다.

응용 소프트웨어를 컴포넌트로 만들기 위한 기술로는 Microsoft의 COM(Component Object Model)[11], Sun의 EJB(Enterprise Java Beans)[12] 그리고 OMG의 CORBA(Common Object Request Broker Architecture)[13] 등이 있다. 이들 세 가지 기술은 효용성, 표준화, 호환성, 성능 등의 관점에서 보편적 기능과 고유의 기능을 갖고 있으며 개발 환경에 따라 그 기술을 선택적으로 사용할 수가 있다. 본 연구의 경우 기존에 구현된 OODesigner가

Microsoft 윈도우 환경의 Visual C++ 언어로 구현되었기 때문에 컴포넌트 기술의 선택 시에 Microsoft 사의 COM을 선택하였다. OLE(Object Linking and Embedding)는 COM에 기초한 구조화된 저장소나 모니커(Moniker) 그리고 통일 데이터 전송(Uniform Data Transfer) 기술들을 조합하여 만들어낸 복합 문서 지원 기술이다.[14] COM/OLE 기술을 통하여 구현된 소프트웨어는 다양한 응용 분야의 문서 작성 소프트웨어간에 삽입(embedding)과 연결(linking)을 통한 협동 작업을 가능하게 하므로 보다 막강한 문서 작성 기능을 제공할 수 있다.

본 논문의 구성은 다음과 같다. 2 장에서는 복합 문서 지원 기능을 OODesigner에 적용하기 위한 설계 내용에 대하여 기술하며 3 장에서는 OLE 컨테이너의 구현 결과를 사용 예와 실행화면 위주로 설명한다. 아울러 4 장에서는 결론과 함께 향후 연구 내용에 대하여 서술한다.

II. 시스템 설계

본 장에서는 OODesigner에 복합 문서 기능을 추가하기 위해 수행된 설계 내용에 대하여 다룬다. 처음으로 복합 문서의 개요에 대하여 개략적으로 알아보고, 다음으로 기존의 OODesigner의 설계에 대해서 MVC(Model View Controller)[15] 패러다임 관점에서 알아본다. 마지막으로 복합 문서 기능을 추가하기 위한 설계 내용을 기술한다. 여기서는 기존의 설계 중에서 변경이 이루어졌던 부분과 새로 추가된 부분을 구분하여 기술한다.

1. 관련 연구

본 연구의 필요성을 확인하기 위하여 기존에 사용되고 있는 객체지향 CASE 도구들에 대한 조사를 실시하였다. IBM 사의 Rational Rose, 볼랜드 사의 Together, 플라스틱 소프트웨어 사의 Agora Plastic, Gentleware 사의 Poseidon for UML, Sparx Systems 사의 Enterprise Architect, Artiso 사의 Visual Case, Visual Paradigm 사의 Visual Paradigm for UML, Aonix 사의 Ameos와 같은 도구들을 조사해 본 결과 이들 모두 OLE 컨테이너 기능과 서버 기능이 존재하지 않았다. 본 논문의 결과로 구현된 OODesigner 만이 OLE 기능을 제공한다.

2. OLE 복합 문서

복합문서는 한 문서 내에 다른 응용 소프트웨어에서 사용하는 정보들을 포함(Embedding)하거나 또는 정보들의 링크를 포함(Linking)할 수 있는 문서이다[14]. OLE는 Microsoft 에서 개발된 복합문서에 대한 표준 기본 틀로서, 복합문서를 작성하고 표시하는 API들의 집합을 정의한다. OLE 지원 소프트웨어는 COM에 기초한 여러 기술인 구조화된 저장소, 모니커, 통일 데이터 전송 기술에서 정의되어 있는 인터페이스들의 함수들을 구현해야 한다. 서로 다른 두 개의 소프트웨어에서 나온 정보가 하나의 복합 문서 안에서 합쳐질 때 소프트웨어의 한 부분은 컨테이너로, 다른 부분은 서버로서의 역할을 각각 맡는다. 개개의 소프트웨어는 그 응용 목적에 따라 컨테이너로서만 작동하거나 혹은 서버로서만 작동하거나 혹은 컨테이너이자 서버로 동시에 작동하도록 구현된다.

3. MVC 관점의 개략 설계

OODesigner는 MVC 패러다임을 기반으로 설계되었다. 그림 1은 OODesigner의 패키지 구조를 보여주는 개략 설계로서 크게 사용자 인터페이스 관련 패키지와 모델 관련 패키지로 설계되었다. 그림 1에서 사용자 인터페이스 관련 패키지는 Controller 기능을 담당하는 프레임 관련 클래스들, 대화상자 및 컨트롤 클래스들을 위한 패키지와 View 기능을 담당하는 뷰 관련 클래스들을 위한 패키지로 구성된다. 모델 관련 패키지는 Model 기능을 담당하는 도큐먼트 관련 클래스들, 표기법 관련 클래스들과 라이브러리 클래스들로 구성된다.

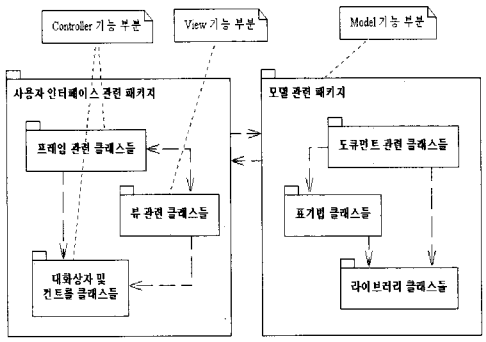

그림 1 OODesigner의 시스템 개략 설계
Fig. 1 Overall Design of OODesigner

OODesigner는 약 200개의 클래스들로 구성되었다. 그러나 그림 1와 같은 MVC 패러다임에 따른 구조 설계에 따라 클래스 간의 결합도가 느슨하게 구현되었기 때문에 시스템의 보수유지가 용이하게 개발되었다. 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. 그림 2는 OLE 기능 추가 이전에 설계되었던 모델 기능 부분의 클래스 다이어그램이다.

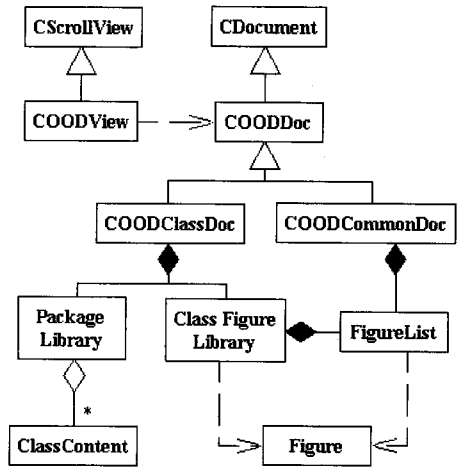

그림 2 모델 관련 클래스 설계
Fig. 2 Class Diagram corresponding to Model part of OODesigner

그림 2에 나타나있는 클래스들의 기능은 다음과 같다.

- COODView: MFC의 CScrollView 클래스로부터 상속되며 OODesigner에서 사용되는 모든 다이어그램들을 보여주는 클래스들의 상위 클래스로서 GetDocument() 함수를 이용하여 도큐먼트의 모델 정보를 가져온 후 화면에 디스플레이 하는 기능을 갖는다.
- COODDoc, COODClassDoc, COODCommonDoc: OODesigner에서는 UML 다이어그램들의 특성을 고려하여 두 가지 종류의 도큐먼트를 분리해서 관리한다. 그 하나인 COODClassDoc 클래스는 UML 클래스 다이어그램을 위한 것이고 다른 하나인 COODCommonDoc 클래스는 클래스 다이어그램을 제외한 유즈 케이스 다이어그램, 시퀀스 다이어그램, 상태 다이어그램 등을 위한 것이다. 도큐먼트의 종류를 두 가지로 구분한 이유는 클래스 다이어그램의 경우는 논리적으로 정보 저장소의 역할을 하기 때문에 그림 정보와 클래스 내용에 대한 정보를 복합적으로 관리해야 하기 때문이고 나머지 다이어그램의 경우는 단순히 그림 정보만을 관리해도 되기 때문이다.
- PackageLibrary, ClassContent: 이들 클래스들은 클래스 다이어그램들을 위한 논리적인 정보 저장소 역할을 한다. 즉 그림의 좌표와 관계없이 도구 상에서 명시된 각 클래스의 명세에 대한 내용을 저장하는 클래스가 ClassContent이며 이들 객체를 전체적으로 관리하는 객체가 PackageLibrary이다.
- ClassFigureLibrary, FigureList: 이들은 다이어그램에 나타나 있는 UML 표기법 들의 좌표 값과 관계되는 그림 정보들을 관리하는 클래스들로서 ClassFigureLibrary는 도구 상에서 여러 개의 뷰로 작성되는 여러 패키지들에 속하는 클래스 다이어그램 정보를 관리하기 위해 여러 개의 FigureList 객체를 이용한다.
- Figure: 이 클래스는 UML에서 사용되는 모든 표기법 객체들을 구현하는 클래스들의 최상위 클래스이다. 이 클래스의 하위 클래스에는 선, 점, 사각형, 삼각형 등 기본 표기법의 조합으로 구성되는 UML 표기법들이 존재한다.

4. 컨테이너 기능 설계

OLE 컨테이너 기능 추가를 위해 핵심적으로 변경된 설계 사항은 그림 3 과 같다. 기존의 클래스들 중에서 수정이 이루어진 클래스들은 COODView 클래스와 COODDoc 클래스이고, 새로 추가된 클래스는 COODCntrlItem 클래스이다.

COODDoc 클래스의 수정된 사항은 다음과 같다. COODDoc 클래스의 상위 클래스가 CDocument 클래스로부터 COleDocument 클래스로 바뀌었는데 그 이유는 OLE 기능의 구현 시에 구조화된 저장소(storage)를 사용하기 때문이다. 그리고 COODDoc 클래스에서 모델 정보를 저장하거나 읽어오는 기능을 하는 함수들인 Serialize(), OnOpenDocument(), OnNewDocuemnt()가 OLE 객체들의 리스트를 관리하며 구조화된 저장소를 처리할 수 있도록 변경되었으며, 루트 저장소를 생성하거나 초기화시키기 위하여 SetRootStg(), OnNewEmbedding(), OnOpenEmbedding() 과 같은 함수들이 추가되었다.

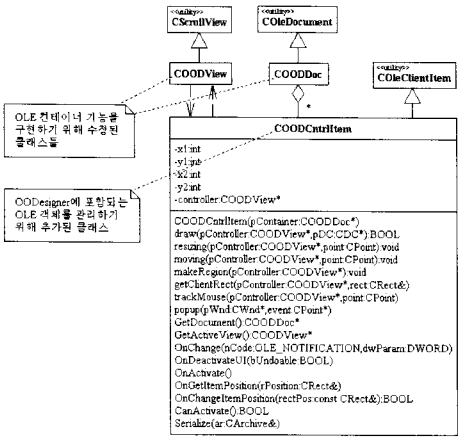

그림 3 컨테이너 기능 설계
Fig. 3 Class Design for Container functionality

COODView 클래스의 수정 사항은 다음과 같다. COODView 클래스에는 OLE 객체를 생성하고 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.

새로 추가된 COODCntrlItem 클래스는 OODesigner에 삽입되거나 연결되는 OLE 객체를 관리하기 위한 목적을 갖는다. 이 클래스의 기능은 삽입된 OLE 객체의 위치와 크기를 관리하고 필요한 경우 디바이스 컨텍스트에 그 객체를 그리거나 객체를 활성화하는 것이다. 그림 3의 COODCntrlItem 내용은 전체 구현 내역 중에서 핵심적인 멤버들만을 제시한 것으로 대문자로 시작되는 함수 이름들은 MFC에서 제공되는 오버라이드된 함수들이고 나머지 멤버들의 의미는 멤버 이름을 통하여 그 역할을 이해할 수 있다.

마지막으로 이루어진 변경사항에는 OLE 컨테이너 기능의 추가를 위하여 메인 클래스인 COODApp에 OLE 라이브러리를 초기화시키기 위한 코드 추가가 있었으며, 프레임 관련 클래스인 CMainFrame 클래스에 OLE 객체를 위한 메뉴 기능의 추가가 있었다.

현재까지 기술된 바와 같이 OLE 기능 추가 과정에서 기존에 존재하던 200 여개의 클래스 중 적은 수의 클래스들에 대해서만 수정이 필요하였는데 이는 기존에 구현된 클래스들이 MVC 패러다임 관점에서 역할 분담이 잘 분산되도록 설계되었던 점에 기인한다.

5. 서버 기능 설계

OLE 서버 기능은 OODesigner 도큐먼트가 훈글이나 MS 워드같이 OLE 컨테이너 기능을 갖는 다른 문서에 삽입되거나 링크되는 것을 가능하게 하기 위한 것이다. 서버 기능의 구현을 위해 설계된 클래스 다이어그램은 그림 4와 같다.

그림 4에서와 같이 서버 기능의 설계를 위해 COODView 클래스와 COODDoc 클래스에 약간의 수정이 이루어졌으며 새로운 클래스 COODCommonView, CInPlaceFrame과 COODSrvrItem이 추가되었다.

COODView 클래스와 COODDoc 클래스에서 수정된 내용은 OODesigner 도큐먼트가 컨테이너 소프트웨어에 삽입되어 도큐먼트가 활성화 될 경우 그에 해당하는 서버 객체를 생성하고 초기화 시키는 기능에 대한 것이다. 추가된 클래스 중에서 COODCommonView의 기능은 OODesigner가 서버로 작동될 때 다이어그램 뷰의 기능을 제공하기 위한 것으로 COODView에 존재하는 모든 이벤트 처리함수 인터페이스를 갖고 있으며 실제 작업은 COODView에 속하는 이벤트 처리 함수로 작업을 위임함으로써 이루어지도록 설계하였다. CInPlaceFrame 클래스의 역할은 삽입된 도큐먼트 객체가 제자리 활성화(in-place activation) 되는 경우 기존 컨테이너의 풀다운 메뉴를 OODesigner 용 풀다운 메뉴로 대체하기 위한 것이다. 마지막으로 COODSrvrItem 클래스는 OODesigner 도큐먼트가 서버 객체로서 작동하게 하는 인터페이스를 제공하는데 이 클래스는 OnDraw() 함수를 이용하여 삽입된 문서를 컨테이너 객체에 그려주는 일을 담당한다.

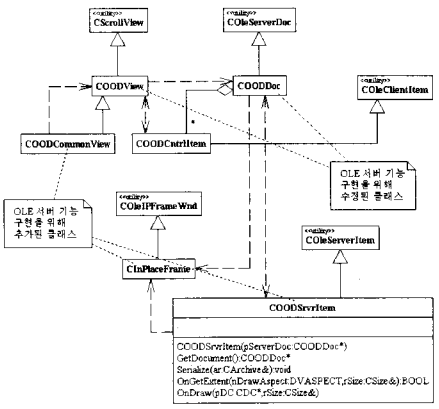

그림 4 서버 기능의 설계
Fig. 4 Class Design for Server functionality

III. 시스템 구현 결과

OODesigner에 OLE 컨테이너/서버 기능을 구현함으로써 OODesigner는 오디오, 비디오 등을 이용한 문서화가 가능해졌으며 아울러 Window에서 제공되는 그림판, 엑셀, 그래픽 편집기 등의 객체를 UML 문서에 연결할 수 있게 되었다. 따라서 OODesigner는 음성, 동영상, 그림, 워드 문서를 편집할 수 있는 소프트웨어와 함께 실행되면서 소프트웨어 개발자에게 막강한 문서화 환경을 제공할 수 있다. 본 절에서는 OLE 기능의 구현 결과를 사용 예와 실행화면 중심으로 제시한다.

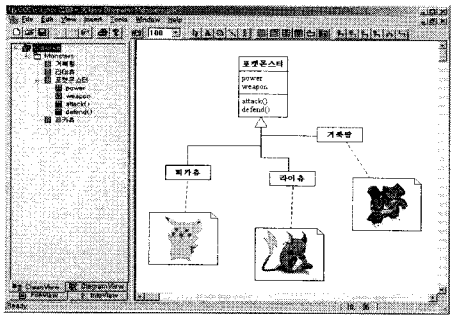

그림 5 UML 문서에 그림 객체들이 삽입된 예
Fig. 5 Screen shot of OODesigner Embedding figure objects

OLE 컨테이너 기능을 통해 제공된 기능 중의 하나는 그림 5 에서와 같이 UML 문서에 그림 파일을 삽입하는 것이다. 그림 5는 포켓몬스터 클래스들을 설계하면서 포켓몬스터 객체가 어떻게 생긴 것인가를 쉽게 설명해 주고 있다.

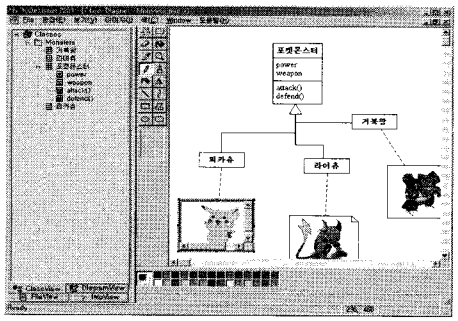

그림 6 그림 객체의 제자리 활성화 예
Fig. 6 In-place activation of embedded figure objects

그림 6의 내용은 그림 5의 문서에 삽입되어 있는 그림 객체를 제자리 활성화(in-place activation) 방식으로 활성화하는 상황을 보여 준다. OODesigner의 사용자는 그림 객체 위에서 마우스를 더블 클릭함으로써 제자리 활성화를 수행할 수 있다. 이 그림에서 볼 수 있다시피 제자리 활성화가 이루어지고 난 다음에 도구 프레임의 메뉴나 툴바, 상태바 등이 그림판을 위한 것으로 교체되었다. OODesigner는 제자리 활성화를 이용한 객체의 편집 기능 외에 오픈 방식의 객체 활성화 기능도 제공하며 특히 OODesigner에 포함된 객체가 삽입이 아니라 링크되어 있는 경우에는 무조건 오픈 방식으로 활성화된다.

OLE 컨테이너 기능을 통해 제공된 다른 기능은 녹음기나 엑셀과 같은 도구를 이용하여 작성된 객체를 삽입하는 것이다. 그림 7의 화면에는 두 가지의 OLE 객체가 삽입되어 있는데 그 하나는 우측 상단에 스피커 형태의 아이콘으로 표시된 음성 객체이며 다른 하나는 우측하단에 있는 엑셀로 작성된 그래프 객체이다. 그림 7의 화면은 녹음기로 작성된 보이스 도큐먼트가 실행되는 상황을 보여준다. 현재 대부분의 CASE 도구가 텍스트 기반의 문서화 기능만을 제공하고 있는 점에서 볼 때 음성을 이용한 문서화는 소프트웨어 문서의 이해력과 사용용이성을 매우 높여줄 수 있을 것이다. 녹음기를 이용한 보이스 객체 경우 뿐 아니라 캠코더로 작성된 비디오 도큐먼트를 삽입하는 것도 가능하며 비디오 영상을 이용한 설계 문서의 작성은 텍스트 위주의 문서 작성이 갖는 표현력의 한계를 극복하는데 큰 도움이 된다.

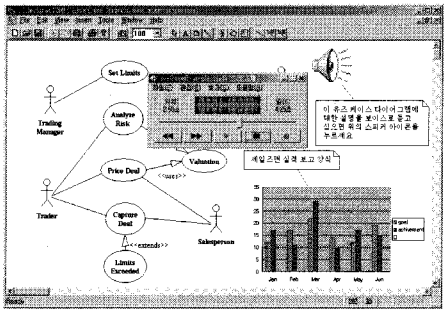

그림 7 음성을 이용한 문서화의 예
Fig. 7 Screen shot of OODesigner Embedding voice objects

마지막으로 보여 줄 구현 결과는 서버 기능에 관한 것이다. 일반적으로 목적 시스템의 설계와 구현을 위해 작성된 UML 다이어그램은 최종보고서나 매뉴얼과 같은 문서에 포함된다. OLE 서버 기능이 지원되지 않는 경우에 보고서 작

성자는 CASE 도구의 화면을 캡처하여 이미지 파일을 만든 다음 그 그림을 보고서에 삽입한다. 그러나 이렇게 작업하게 되면 원본 설계 문서에 변경이 생길 때마다 화면 복사 작업을 반복해야만 하며 설계 문서와 보고서 간의 일관성을 유지 할 수 없다. 반면에 OODesigner 경우와 같이 CASE 도구가 OLE 서버 기능을 지원하게 되면 삽입이나 링크 방식으로 UML 문서와 보고서를 연동시켜 작업할 수 있으므로 문서화의 편의성을 도모할 수 있을뿐더러 설계문서와 보고서간의 일관성 유지를 달성할 수 있다. OODesigner 설계 문서는 파워포인트, MS 워드, 훈글과 같이 OLE 컨테이너로 작동될 수 있는 모든 소프트웨어에 삽입될 수 있으며 그림 8의 사용 예는 OODesigner 설계 문서가 훈글에 포함된 상태에서 제자리 활성화가 이루어지고 있는 상황을 보여준다.

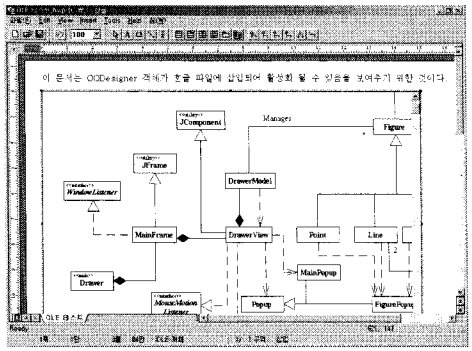

그림 8 훈글 문서에 포함된 OODesigner 객체의 제자리 활성화 예
Fig. 8 Screen shot of OODesigner running as a server within a word processor

IV. 결론

재사용성 확보를 위한 복합문서 지원 기술이 보편화되어 있음에도 불구하고 현재 사용되고 있는 대부분의 CASE 도구에는 이러한 기술을 지원하고 있지 못하다. CASE 도구에 컴포넌트 기술을 도입하여 복합문서 기능을 지원함으로서 얻을 수 있는 장점은 다음과 같다.

- 컴포넌트 기술을 이용함으로서 UML문서에 OLE 객체를 저장하고 재생하는 기능을 CASE 도구 내에서 구현하지 않고 기존의 멀티미디어 실행 프로그램을 연결해서 사용할 수 있다.
- UML 설계문서 작성 시 이미지, 사운드, 동영상을 이용하여 설계문서를 편리하고 자연스럽게 설명 가능함으로 소프트웨어개발자에게 강화된 문서화 환경을 제공할 수 있다.

본 논문에서는 기존에 개발되었던 CASE 도구 OODesigner에 복합문서 지원 기능을 적용하기 위한 목적으로 OLE 컨테이너 및 서버 기능을 추가로 개발한 연구 내용을 기술하였다. 본 논문에서 논의된 주된 내용은 OLE 기능의 추가를 위한 설계 결과와 도구의 유용성을 보여주는 구현 결과에 대한 것이다. 현재 OODesigner에 대한 테스트를 수행 중이며 시스템의 안정성이 확보되면 UML 모델링을 위한 유용한 도구로서의 역할을 할 수 있을 것이다. 본 기능의 구현 이후에 이루어져야 할 연구 과제는 OODesigner가 상업적 CASE 도구들과 경쟁할 수 있도록 추가적으로 기능을 보강하는 것이다.

참고문헌

[1] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorenzen, Object-Oriented Modeling and Design, Prentice Hall, 1991.

[2] P. Coad, E. Yourdon. Object-Oriented Analysis, Yourdon Press, 1990.

[3] R. J. Wirfs-Brock, R. E. Johnson, "Surveying Current Research in Object-Oriented Design," Communications of the ACM, 33(9), pp. 104-124, September 1990.

[4] M. Fowler, K. Scott, UML Distilled: Applying the Standard Object-Oriented Modeling Language, Addison-Wesley, 1997.

[5] http://www-306.ibm.com/software/rational

[6] http://www.borland.com/us/products/together/index.html

[7] 조장우, 김태균, "객체지향 CASE 도구에 대한 재구조화 실험", 한국정보처리학회 논문지, 제6권, 제4호, pp.932-940, 1999.

[8] 홍의석, 김태균, "객체 지향 CASE 도구 OODesigner의 플랫폼 이식 사례 연구," 한국정보처리학회 논문지, 제7권, 제9호, pp. 2857-2866, 2000.

[9] D. Teichrow, E. Hershey, "PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems," Trans. Software Eng.,

Vol. SE-3, No. 1, January 1977
[10] M. Alford, "A Requirement Engineering Methodology for Real-Time Processing Requirements," Trans. Software Eng., Vol. SE-3, No. 1, January 1977
[11] Richard Grimes, et al., Beginning ATL COM Programming, Wrox, 1999.
[12] E. Roman, Mastering Enterprise Java Beans, John Wiley & Sons, 1999.
[13] T. J. Mowbray, R. Zahavi. The Essential CORBA, John Willey & Sons, 1995.
[14] D. Chappell. Understanding ActiveX and OLE, Microsoft Press, 1997.
[15] Y. P. Shan, "An Event-Driven Model-View-Controller Framework for Smalltalk," In Proceedings of OOPSLA89, pp. 347-352, New Orleans, USA, October 1989.

저 자 소 개

조 장 우

1992 서울대학교 계산통계학과 (학사)
1994 서울대학교 전산과학과 (석사)
2003 한국과학기술원 전산학과 (박사)
현재 동아대학교 컴퓨터공학과 부교수
관심분야: 프로그램 분석, 임베디드 시스템, 객체지향프로그래밍

김 태 균

1985 서울대학교 자연과학대학(학사)
1987 서울대학교 전산과학과 (석사)
1995 서울대학교 전산과학과 (박사)
현재 부산외국어대학교 컴퓨터공학부 교수
관심분야 : 소프트웨어공학, 객체지향 방법론, 개발환경, CASE도구