웹에서 이벤트 다이어그래머 애플릿의 설계 및 구현

반 종 오† · 최 형 진††

요 약

HTTP의 등장으로 전세계적으로 인터넷붐이 조성되고 있는 가운데 플랫폼과 독립적으로 실행되고 동적 문맥을 제공하는 자바 기술이 등장하였다. 본 논문에서는 이러한 변화에 따라 웹에서 사용가능한 케이스 서버의 필요성을 제시하였고 웹과 자바 기술을 사용하여 이 케이스 서버에 장착할 수 있는 이벤트 다이어그래머 애플릿을 설계하고 구현하였다.

이벤트 다이어그래머는 케이스 도구를 구성하는 여러 부품중 하나이다. 즉, 이벤트 다이어그래머는 이벤트와 작용의 관계를 표현하는 그래픽 표현 요소들로 구성된 표기법인 이벤트 다이어그램을 작성하는 도구이다. 이 논문에서 작성한 이벤트 다이어그래머 애플릿은 웹에서 Martin/Odell의 객체지향 방법론에 따르는 이벤트 다이어그램을 작성해 주는 도구이다.

Design and Implementation of Event Diagrammer Applet in the Web

Ban Jong Oh† · Choi Hyung Jin††

ABSTRACT

In Internet environments, a Java technique which is platform independent and supports interactive content on a Web pages became popular with increasing Internet concerns. This paper shows needs of CASE servers executable on the Web and contains design and implementation of event diagrammer applets with the Web and Java techniques, which are attachable to the CASE servers.

Event diagrammer is one of components which compose CASE tools and a tool which draw event diagrams. The EDA(Event Diagrammer Applet) in this paper support object-oriented methodology of Martin/Odell and a tool which draw event diagrams in the Web.

1. 서 론

객체지향 소프트웨어 개발 방법론은 객체지향 분석과 설계로부터 시작되며, 복잡한 소프트웨어를 구현할 때 개발자와 일반 사용자의 이해를 위하여 완전하고 명료한 다이어그램 표현이 필수적이다. 이러한 다이어그램 표현을 위하여 수 많은 객체지향 분석 및 설계 방법론이 대두되고 있으며 각 방법론을 지원하는 OCASE(Object-oriented Computer Aided Software Engineering) 도구가 요구되고 있다[1][5][13].

또한 최근에 HTTP(Hyper Text Transfer Protocol)가 등장하면서 웹 환경이 급격히 확장되고 있다. 그리고 웹 환경에서 플랫폼과 독립적으로 실행되는 소프트웨어를 개발할 수 있는 자바(Java) 언어가 등장하였다. 앞으로는 웹과 자바 기술이 결합함으로써

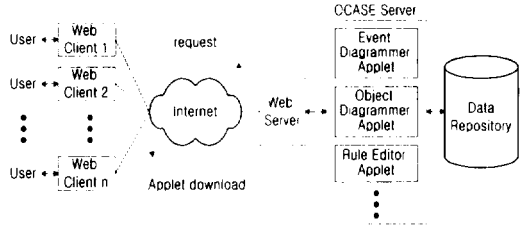
† 준 회 원 : 한림전문대학 인터넷정보과
†† 종신회원 : 강원대학교 전자계산학과

논문접수 : 1997년 6월 21일, 심사완료 : 1998년 2월 4일

기존에 소프트웨어 패키지를 구입하여 사용하는 방식에서 웹 페이지에 장착된 자바 애플릿을 계약 임대(Transaction- Oriented Rental)하여 사용하는 새로운 방식으로 전환될 것이다[7][8][10][11].

현재 많은 인터넷 지원 응용 프로그램이 개발되고 있으며 이러한 개발 노력이 증가할수록 인터넷을 지원하는 OCASE 도구 즉, OCASE 서버와 같은 더욱 강력한 도구가 필요하게 될 것이다. 그러나 현재 인터넷을 지원하는 도구는 간단한 문서 편집기 수준에 불과하다[10].

따라서 웹과 자바 기술을 사용하여 다이어그램 표현과 인터넷을 지원하는 OCASE 서버를 자바 애플릿으로 구현하여 (그림 1.1)과 같이 웹 서버에 연결하면 웹 클라이언트마다 OCASE 도구를 저장하지 않아도 필요할 때마다 OCASE 서버의 OCASE 도구를 호출하여 사용할 수 있게 된다. 예를들어 많은 응용 프로그램 개발자가 OCASE 도구를 사용하는 기업체의 경우, 이렇게 함으로서 동일한 OCASE 도구를 모든 응용 프로그램 개발자의 컴퓨터마다 중복해서 설치하지 않고 인터넷상의 웹 서버에 연결된 OCASE 도구를 이용함으로써 비용과 저장 공간의 사용을 최소화 할 수 있다. 그리고 OCASE 도구를 개발한 개발자의 입장에서는 OCASE 도구의 유지보수 비용을 줄일 수 있게 된다.



(그림 1.1) OCASE 서버
(Fig. 1.1) OCASE server

위와 같은 하나의 완전한 OCASE 도구를 설계하고 구현하는 것은 많은 노력과 시간을 필요로 하므로 본 논문에서는 OCASE 도구를 구성하는 여러 부품중에서 우선적으로 구현되어야 할 것을 하나만 선정하여 그 부품에 대하여 설계하고 구현한다.

현재 다이어그램 표현을 지원하는 OCASE 부품 중에는 이벤트 다이어그래머, 객체 다이어그래머, 규칙 에디터 등이 있지만 대부분의 소프트웨어들은 윈도스 95, 윈도스 NT, 매킨토시, 솔라리스 등의 그래픽 사용자 인터페이스(GUI : Graphical User Interface) 환경에서 마우스 객체 등의 객체에서 발생하는 이벤트를 처리해 주는 것들이다.

따라서 본 논문에서는 현재의 소프트웨어 개발 추세에 맞추어 객체지향 분석과 설계시에 이벤트와 작용의 설계를 쉽게 할 수 있는 이벤트 다이어그래머를 우선적으로 자바 애플릿으로 설계하고 구현한다.

2장에서는 OCASE와 관련된 연구를 알아보고 본 논문에서 사용할 OCASE 방법론을 선정한다. 3장에서는 EDA(Event Diagrammer Applet)를 3계층으로 나누어 설계하고 4장에서는 3장에서 설계한대로 자바 언어를 사용하여 EDA를 구현한다. 마지막으로 5장에서는 결론 및 향후 연구과제를 기술한다.

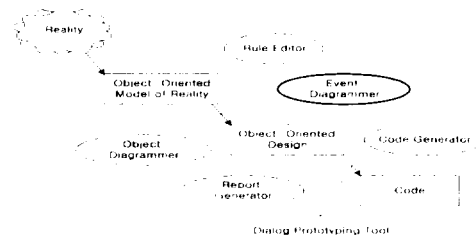
2. 관련 연구

2.1 객체지향 방법과 OCASE 도구

객체지향 방법이란 주어진 문제를 객체지향 분석(OOA : Object-Oriented Analysis) 및 설계(OOD : Object-Oriented Design)를 하여 이를 프로그램으로 구현하는 방법을 말한다.

현재 객체지향 분석 및 설계 방법은 OCASE, 코드 생성기, 비쥬얼 프로그래밍, 클라이언트/서버, 클래스 라이브러리, 객체지향 데이타베이스 등의 여러 분야에서 사용되고 있다[1][2][3][4][5].

객체지향 CASE 도구는 객체지향 방법을 자동화시키는 것으로 (그림 2.1)와 같이 실세계로부터 코드까지의 여러 단계에 걸쳐 사용된다[1][5]. 그 중에서 이벤트와 작용의 관계를 잘 표현하는 이벤트 다이어그램을 쉽게 작성할 수 있는 이벤트 다이어그래머에 초점을 맞추었다.



(그림 2.1) 객체 지향 CASE 도구
(Fig. 2.1) Object-oriented CASE tools

〈표 2.1〉 객체지향 분석 및 설계 방법[6]
〈Table 2.1〉 Object-oriented analysis and design methodology[6]

Author	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Coad-Yourdon	y	y	y	y	y	y	y	n	n	y	n	n	n	n
Martin-Odell	y	y	y	y	y	y	y	y	y	y	y	y	n	n
Raumbaugh	y	y	y	y	y	y	y	y	n	y	y	y	y	y
Shlaer/mellor	y	y	y	n	y	y	y	y	y	y	y	n	y	n
Wirfs-Brock	y	y	y	y	y	n	n	n	n	n	?	n	n	n

1. inheritance of attributes
2. multiple inheritance
3. attributes
4. IS-PART-OF
5. relation and function
6. transition diagram
7. event
8. event diagram
9. trigger
10. data flow diagram
11. concurrency
12. object concurrency
13. supporting tool
14. integrated tool

〈표 2.1〉의 여러 방법들 중에서 EDA 설계와 구현에 필요한 이벤트와 작용에 대해 잘 정의되어 있는 것은 Martin/Odell 방법과 Shlaer/Mellor 방법이다. 그러나 Shlaer/Mellor 방법은 객체지향 방법보다 전통적인 구조적 방법과 정보 모델의 개념 적용에 더 많은 비중을 두고 있다[6][9][12]. 따라서 본 논문에서는 Martin/Odell의 방법을 사용한다.

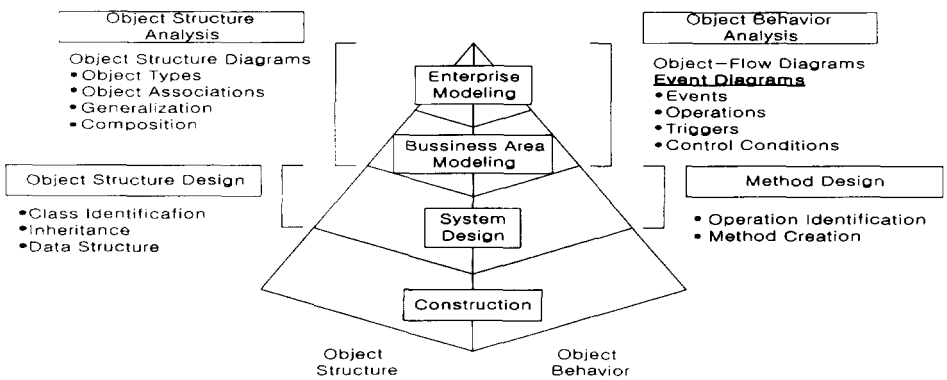
현재 객체지향 방법을 지원하는 CASE 도구[12] 중에서 Martin/Odell 방법을 지원하는 것은 Associative Design Technology사의 Ptech와 Intellicorp사의 OMW 등이 이벤트 다이어그래머를 제공한다. 그러나 Ptech는 유닉스만 지원하며 OMW는 유닉스와 윈도스만을 지원하고 있다. 그리고 두 도구 모두 인터넷을 지원하지 못한다.

따라서 본 논문에서는 Martin/Odell 방법을 사용하여 멀티 플랫폼과 인터넷을 지원하는 이벤트 다이어그래머 애플릿(EDA)을 설계하고 구현한다.

2.2 Martin/Odell의 이벤트 다이어그래머[1][2]

Martin/Odell의 방법은 (그림 2.2)와 같으며 이 방법을 지원하는 다이어그래머에는 이벤트 다이어그래머, 객체 다이어그래머, 규칙 에디터 등이 있지만 본 논문에서는 EDA 설계와 구현에 필요한 이벤트 다이어그램을 작성하는 도구인 이벤트 다이어그래머만을 다루기로 한다.

이벤트 다어어그램은 〈표 2.2〉와 같이 작용



(그림 2.2) Martin/Odell 방법
(Fig. 2.2) Martin/Odell method

(operation), 외부 작용(external operation), 트리거(trigger), 이벤트 분할(event partition), 제어 조건(control condition), 이벤트(event), 선행 조건(prerequisite), 타임 이벤트(time event) 등의 8가지 그래픽 표현 요소로 구성된다.

여기서 작용은 상태 변화를 수행하는 과정으로 객체지향 프로그래밍에서 메소드로 정의된다. 외부 작용은 시스템 외부의 이벤트를 발생시키는 작용을 의미한다. 트리거는 이벤트와 작용간의 일시적인 관계를 의미하며 이벤트가 발생했을 때 연결된 작용을 호출한다. 이벤트 분할은 하나의 이벤트가 둘 이상의 이벤트로 분할될 때 사용한다. 제어 조건은 어떤 작용이 시작되는데 필요한 상태를 확인한다. 이벤트는 작용에 의한 상태 변화를 정의한다. 선행 조건은 이벤트와 작용간의 특정한 관계가 존재하는 것이 아니라 단지 특정한 작용을 시작하기 위하여 먼저 충족되어야 할 조건을 의미하며 타임 이벤트는 어떤 작용이 발생할 시간이 되었음을 알려주는 이벤트이다. 이러한 그래픽 표현 요소들로 구성된 표기법을 이벤트 다이어그램이라고 한다

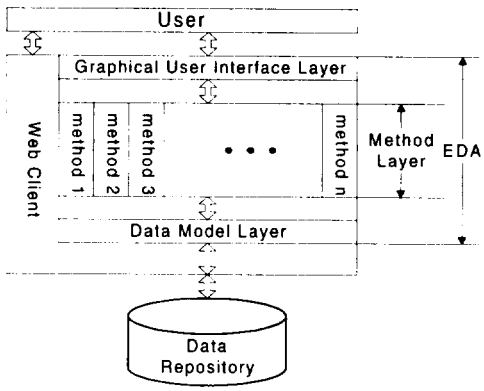
〈표 2.2〉 EDA의 그래픽 표현 요소
〈Table 2.2〉 EDA's graphical elements

Operation	Operation results in events. Event cause other operations to occur
External Operation	Operations that cause events are external to the system
Trigger	The casual relationship between event and operation
Event Partition	Events are subtyped
Control Condition	Operation requires multiple triggers to invoke it
Event	A noteworthy change in the state of an object
Prerequisite	No relationship between event and operation
Time Event	A clock time being reached that triggers some operation

3. EDA의 설계

본 논문에서는 Martin/Odell 방법을 사용하여 유닉스, 윈도스 95, 매킨토시 등의 플랫폼을 지원하고 인터넷 상에서 웹 서버에 장착 가능하도록 EDA를 설계한다.

(그림 3.1)은 인터넷을 통해 OCASE 서버로부터 웹 클라이언트로 다운로드된 EDA의 구조이다. EDA는 그림과 같이 사용자와의 대화를 담당하는 그래픽 사용자 인터페이스 계층, EDA의 기능을 수행하는 메소드 계층, 그리고 메소드에서 처리된 데이터를 저장하는 데이터 모델 계층 등의 세 개의 계층으로 설계한다. 본 논문에서는 EDA를 사용하여 작성한 이벤트 다이어그램을 통신 트래픽을 줄이기 위하여 클라이언트측의 데이터 저장소에 저장한다.



(그림 3.1) EDA의 구조
(Fig. 3.1) Structure of EDA

3.1 그래픽 사용자 인터페이스 계층

EDA의 그래픽 사용자 인터페이스 계층은 사용자와 메소드 계층간의 대화를 담당하며 이 대화 기능에는 이벤트 다이어그램을 작성하고 수정하고 저장하는 기능, 저장된 이벤트 다이어그램을 불러오는 기능, 이벤트 다이어그램을 기본적인 자바 또는 C++ 골격 코드로 변환하는 기능과 사용자에게 도움말을 제공하는 기능 등

을 포함하도록 설계한다. 이러한 기능을 수행하기 위하여 메뉴는 크게 파일(File)과 도움말(Help)을 포함한다. 파일 메뉴에는 새로운 이벤트 다이어그램 작성하기(New), 이벤트 다이어그램 읽기(Open), 이벤트 다이어그램 저장하기(Save), 이벤트 다이어그램을 자바 골격 코드로 변환하기(Java skeleton code), 이벤트 다이어그램을 C^{++} 골격 코드로 변환하기(C^{++} skeleton code), 종료(Exit) 기능 등을 포함하도록 설계한다. 이외에도 그래픽 사용자 인터페이스에서는 이벤트 다이어그램을 전부 삭제하기(Clear all), 객체를 이동하기(Move), 객체를 삭제하기(Delete), 객체를 복사하기(Copy) 기능을 지원하고 〈표 2.2〉과 같이 Martin/Odell 방법론에서 지원하는 작용, 외부 작용, 트리거, 이벤트 분할, 제어 조건, 이벤트, 선행 조건, 타임 이벤트 등의 8가지 그래픽 표현 요소를 입력할 수 있는 기능을 포함하도록 설계한다.

3.2 메소드 계층

EDA의 메소드 계층에서는 〈표 2.2〉과 같이 8가지 그래픽 표현 요소를 입력할 수 있다. 그래픽 표현 요소 중 작용의 기능을 수행할 작용 클래스는 (그림 3.2)와 같이 설계하였다. 나머지 클래스들도 작용 클래스와 같이 자바 API의 Object 클래스를 상속하여 설계하였으며 클래스 선언부, 변수 선언부, 생성자 선언부, 메소드 선언부, 클래스 종료부 등으로 구성된다. 각 그래픽 표현 요소들의 기능을 수행하는 클래스들은 각각의 고유 기능을 수행할 메소드, 자신의 고유 정보를 저장할 변수, 나중에 골격 코드 생성에 이용될 정보 등을 포함한다.

(그림 3.2)의 작용 클래스는 작용의 위치와 크기를 나타내는 x, y, width, height, arcWidth, arcHeight와 작용의 이름을 나타내는 operationName, 작용의 지정자인 modifier1, modifier2, modifier3, 작용의 리턴형인 returnType 등을 정의하고 이벤트가 작용 안에서 발생했는지, 아니면 왼쪽이나 오른쪽 또는 위나 아래에서 발생했는지를 알 수 있도록 inside(), locate_left(), locate_right(), locate_up(), locate_down() 등의 메소드와 작용 객체 생성자를 정의한다.

```
public class Operation extends Object {        // Operation 클래스 선언
    public short x;                             // x 변수 선언
    public short y;                             // y 변수 선언
    public short width;                         // width 변수 선언
    public short height;                        // height 변수 선언
    public short arcWidth;                      // arcWidth 변수 선언
    public short arcHeight;                     // arcHeight 변수 선언
    public String operationName;                // operationName 변수 선언
    public byte modifier1;                      // modifier1 변수 선언
    public byte modifier2;                      // modifier2 변수 선언
    public byte modifier3;                      // modifier3 변수 선언
    public byte returnType;                     // returnType 변수 선언
    public short totalParameterCount;  // totalParameterCount 변수 선언
    public byte parameterCount;                 // parameterCount 변수 선언
    public Operation()                          // Operation 객체 생성자 선언
    public Operation(String operationName, short x, short y, short width,
           short height, byte modifier1, byte modifier2, byte modifier3,
           byte returnType, short totalParameterCount, byte parameterCount)
                                                // Operation 객체 생성자 선언
    public boolean inside(short x, short y)   // inside 메소드 선언
    public boolean locate_left(short x, short y)     // locate_left 메소드 선언
    public boolean locate_right(short x, short y)   // locte_right 메소드 선언
    public boolean locate_up(short x, short y)      // locate_up 메소드 선언
    public boolean locate_down(short x, short y)
                                                // locate_down 메소드 선언
    public void draw(Graphics g, short xOffset, short yOffset)
                                                // draw 메소드 선언
}                                               // Operation 클래스 종료
```

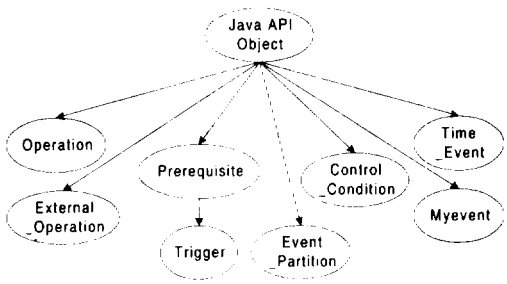
(그림 3.2) Operation 클래스
(Fig. 3.2) Operation class

그리고 작용 객체를 화면에 그려주는 draw() 메소드를 정의한다.

8개의 그래픽 표현 요소에 해당하는 클래스는 자바 API의 Object 클래스를 상속받도록 설계하였다. 특히 이벤트와 제어 조건은 작용, 외부 작용, 타임 이벤트의

상하좌우 4방향에 결합할 수 있으며, 이벤트 분할은 상하좌우의 4방향을 가지도록 설계하였다. 이벤트 다이어그램의 표현 요소 중에서 이벤트의 경우에는 자바 클래스에 같은 이름을 가진 클래스가 존재하므로 이름을 마이이벤트(Myevent)로 명명하여 사용하였다.

위에서 설계한 EDA의 8가지 그래픽 표현 요소들의 클래스 상속 관계는 (그림 3.3)과 같다.



(그림 3.3) EDA 요소 클래스의 상속 관계
(Fig. 3.3) Inheritance relation of EDA element classes

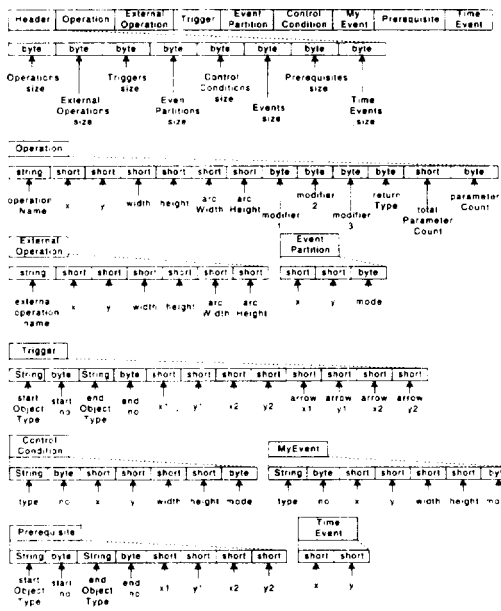
3.3 데이터 모델 계층

EDA는 사용자가 작성한 이벤트 다이어그램을 웹 클라이언트에 저장하여 사용자가 자신이 작성한 이벤트 다이어그램을 직접 관리할 수 있다.

본 논문의 EDA 자료 저장 구조는 (그림 3.4)와 같다. 자료 저장은 각 그래픽 표현 요소의 기능을 수행하는 클래스들의 고유 정보와 골격 코드 생성에 필요한 정보를 저장하도록 설계하였다. 먼저 헤더 부분은 각 그래픽 표현 요소의 수를 저장하며 각 표현 요소의 크기는 -128부터 127까지 표현 가능한 1바이트 정수인 바이트(byte)를 사용한다.

Operation과 External_Operation은 자신의 이름, 시작점, 넓이, 높이에 대한 정보를 저장한다. 특히, Operation의 경우에는 지정자, 리턴형, 파라메터의 수 등의 정보를 byte형으로 저장한다. 여기서 각각의 이름은 String형으로 설계하고 캔버스 크기는 1000×1000 픽셀로 설정하였으므로 시작점, 넓이, 높이는 -32768부터 32767을 표현할 수 있는 short형으로 설계한다.

Trigger, Event_Partition, Control_Condition, Event, Prerequisite, Time_Event의 시작점이나 끝점, 넓이, 높이 등은 캔버스의 크기를 고려해서 short형으로 설계하고 Event_Partition, Control_Condition, Event의 모드는 상하좌우의 4가지이므로 byte형으로 설계한다.



(그림 3.4) EDA의 자료 저장 구조
(Fig. 3.4) Data store structure of EDA

4. EDA의 구현

이 장에서는 앞에서 설계한 EDA를 구현한다. 자바 애플릿은 (그림 1.1)과 같이 자바를 지원하는 브라우저에 의해 네트워크를 통해 다운로드되고 웹 페이지를 통해 실행되는 대화형 프로그램을 의미한다.

본 논문에서는 EDA를 구현하기 위하여 자바 런타임 시스템 환경으로 윈도스 95, 솔라리스 2.5 에서 각각 JDK(Java Developer's Kit)1.1.1를 사용하였다. EDA의 그래픽 사용자 인터페이스는 자바 API의 AWT(Abstract Window Toolkit) 패키지를 사용하여 구현하였다.

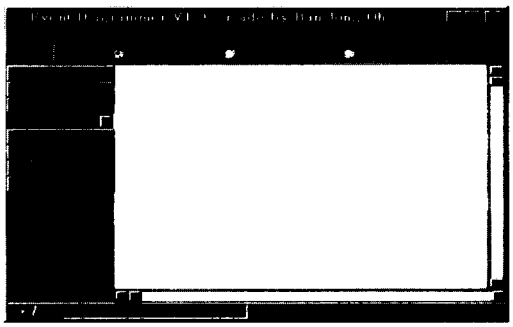
인터페이스는 파일과 도움말 메뉴를 가지고 있으며

파일 메뉴에는 새로 시작하기(new), 열기(open), 저장(save), 자바 골격 코드(java skeleton code), C++ 골격 코드(C++ skeleton code), 종료(exit) 기능을 지원하도록 구현한다. 그러나 본 논문에서 정의한 EDA 다이어그램에 입력된 정보로는 작용의 경우 3.2절에서 설명한 3가지 지정자, 리턴형, 파라메터 등의 정보에 대한 초보적인 골격 코드만 생성할 수 있다.

인터페이스 중앙에는 이벤트 다이어그램을 작성할 수 있도록 AWT의 캔버스(canvas)를 삽입하였다. 캔버스 왼쪽에 8가지 그래픽 표현 요소 중에서 상하좌우의 방향성을 가지는 Event_Partition은 AWT의 초이스(choice)를 사용하여 구현하였으며 나머지는 AWT의 버튼(button)을 사용하여 구현하였다.

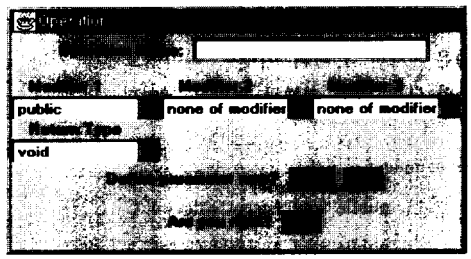
캔버스 위쪽에는 8가지 그래픽 표현 요소를 추가(add), 이동(move), 삭제(delete)할 수 있도록 AWT의 체크박스(checkbox)를 사용하여 구현하였고, 캔버스 내의 모든 객체들을 삭제할 수 있는 전부 삭제(clear all) 버튼을 체크박스 옆에 추가하였다.

(그림 4.1)은 윈도스95에서 호출한 EDA의 그래픽 사용자 인터페이스이다.

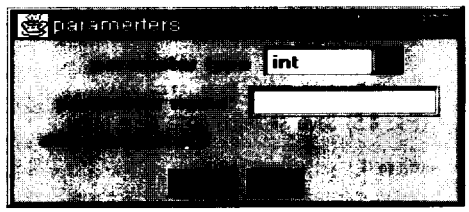


(그림 4.1) EDA 사용자 인터페이스(윈도스95)
(Fig. 4.1) EDA user interface(Windows95)

(그림 4.2)는 이벤트 다이어그램의 구성 요소인 작용을 입력하는 화면으로 작용의 이름, 작용의 지정자, 작용의 리턴형 등을 입력할 수 있다. 작용의 지정자는 public, protected, private, private protected, friendly 등을 선택할 수 있는 Modifier1, static 여부를 선택할 수 있는 Modifier2, abstract, final, native, synchronized 등을 선택할 수 있는 Modifier3 등으로 구성된다. 그리고 리턴형에서도 void, byte, short, int, long, float, double, boolean, char 등을 선택할 수 있다. 그리고 파라메터 정의는 (그림 4.3)과 같은 파라메터 입력 화면을 통하여 byte, short, int, long, float, double, boolean, char 등의 파라메터 형을 선택하거나 파라메터 이름을 입력할 수 있다.

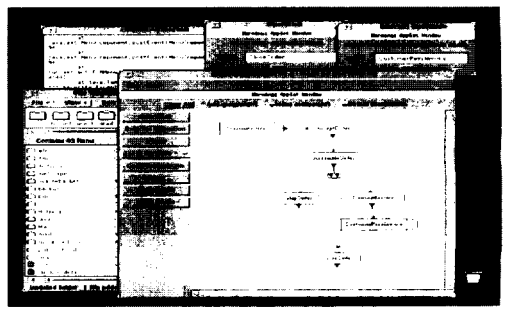


(그림 4.2) 오퍼레이션 입력 다이얼로그
(Fig. 4.2) Operation input dialog



(그림 4.3) 파라메터 입력 다이얼로그
(Fig. 4.3) Parameter input dialog

(그림 4.4)는 구현된 EDA를 솔라리스2.5에서 호출하여 이벤트 다이어그램을 작성한 예이다. 이 화면 위에는 작용과 외부 작용의 이름을 입력하는 프레임이 나타나 있고 EDA의 그래픽 사용자 인터페이스 내에 있는 것은 작성중인 이벤트 다이어그램의 예이다.



(그림 4.4) 솔라리스 2.5에서의 EDA 사용 예
(Fig. 4.4) Example using of EDA in Solaris 2.5

본 논문에서는 EDA로 작성한 이벤트 다이어그램을 기본적인 자바 골격 코드와 C++ 골격 코드로 변환할 수 있는 기능을 지원한다. 사용자는 먼저 EDA에서 이벤트 다이어그램을 작성하거나 이미 저장된 이벤트 다이어그램을 불러온 후 파일 메뉴의 자바 골격 코드, C++ 골격 코드 기능을 사용하여 자바 골격 코드나 C++ 골격 코드를 생성할 수 있다. (그림 4.5)과 (그림 4.6)은 (그림 4.4)의 이벤트 다이어그램을 자바 골격 코드와 C++ 골격 코드로 변환한 예이다.

```
class sample {
        TransmitsOrder( );
        public void AcceptOrder( ) {
        }
        public void AssembleOrder( ) {
        }
        public void ShipOrder( ) {
        }
        public void TrinsmitInvoice( ) {
        }
        public void CloseOrder( ) {
        }
        CustomerPaysInvoice( );
}
```

(그림 4.5) 자바 골격 코드 생성
(Fig. 4.5) Java skeleton code generation

```
class sample {
public:
        sample( );
        ~sample( );
        TransmitsOrder( );
        AcceptOrder( );
        AssembleOrder( );
        ShipOrder( );
        TrinsmitInvoice( );
        CloseOrder( );
        CustomerPaysInvoice( );
}
sample::sample( ) {
}
sample::~sample( ) {
}
sample::AcceptOrder( ) {
}
sample::AssembleOrder( ) {
}
sample::ShipOrder( ) {
}
sample::TrinsmitInvoice( ) {
}
sample::CloseOrder( ) {
}
```

(그림 4.6) C++ 골격 코드 생성
(Fig. 4.6) C++ skeleton code generation

5. 결론 및 향후 연구과제

본 논문에서는 객체지향 방법을 지원하고 인터넷에서 사용할 수 있는 OCASE 도구의 필요성을 제시하였고 그 중에서 우선 객체지향 분석과 설계시에 그래픽 표현 요소를 사용하여 이벤트와 작용의 설계를 쉽게 할 수 있는 이벤트 다이어그래머에 중점을 두었다. 그리고 대표적인 객체지향 방법들 중에서 이벤트 다이어그래머에 필요한 이벤트와 작용에 대하여 잘 정의된 Martin/Odell의 방법을 선택하여 인터넷의 웹 서버에 연결하여 사용할 수 있는 EDA를 설계하고 구현하였다.

본 논문에서 구현한 EDA는 웹과 자바 기술을 사용함으로써 기존의 다이어그래밍 도구들이 인터넷과 멀티 플랫폼을 지원하지 못하는 문제점을 해결할 수 있고 인터넷 상의 특정한 웹 서버에 EDA를 연결시킨 후 다른 웹 클라이언트에서 필요할 때에만 호출하여 사용할 수 있다. 다시말해서 웹 클라이언트마다 EDA를 저장하지 않고 웹 서버의 EDA를 호출하여 사용할 수 있다. 서론에서 언급했듯이 많은 응용 프로그램 개발자가 EDA를 사용하는 기업체의 경우, 이렇게 함으로서 동일한 EDA를 모든 응용 프로그램 개발자의 컴퓨터마다 중복해서 설치하지 않고 인터넷 또는 인트라넷 상의 웹 서버에 연결된 EDA를 이용함으로써 비용과 저장 공간의 사용을 최소화 할 수 있다. 그리고 EDA를 개발한 개발자의 입장에서는 EDA의 유지보수 비용을 줄일 수 있다. 또한 EDA를 구현함으로써 향후 OCASE 서버를 구현하고 NC(Network Computer)에 응용할 수 있는 기초 기술을 획득할 수 있다. 앞으로 더욱 많은 플랫폼이 자바를 지원하게 될 것이므로 본 논문에서 구현한 EDA는 더 많은 플랫폼에서 사용할 수 있게 될 것이다.

EDA를 사용하여 작성한 이벤트 다이어그램을 자바 골격 코드와 C++ 골격 코드로 변환할 수는 있으나 본 논문의 이벤트 다이어그램에 저장된 정보로는 작용의 경우, 3가지 지정자, 리턴형, 파라메터 등의 정보에 대한 기본적인 골격 코드만 생성할 수 있다. 골격 코드 생성은 아직 초보적인 단계이며 앞으로 Martin/Odell 방법론을 지원하는 OCASE 부품들을 연구하여 각 부품에 저장되는 정보들을 골격 코드에 반영하여 더욱 정밀한 골격 코드를 생성하기 위하여 더많은 연구와 노력이 필요하다.

또한 본 논문에서 설계하고 구현한 EDA 외에 클래스 다이어그래머, 규칙 에디터, 코드 생성기 등도 웹과 자바 기술을 이용하여 결과적으로 하나의 완벽한 인터

넷용 OCASE 서버를 구축하는 연구가 필요하다.

참 고 문 헌

[1] J. Martin, "Principles of Object-Oriented Analysis and Disign", Prentice Hall, 1993.

[2] J. Martin, and J. J. Odell, "Object-Oriented Methods", Prentice Hall, 1996.

[3] R. S. Pressman, "Software Engineering-A Practitioner's Approach", Pressman, 1992.

[4] G. Booch, "Object-Oriented Analysis and Desigh with Applications", Benjamin/Cummings, 1994.

[5] D. E. Brumbaugh, "Object-Oriented Development", John Wiley & Son, 1994.

[6] D. Champeauz and P. Farue, "A Comparat-ive Study of Object-Oriented Analysis Methods", Journal of Object-Oriented Programming, 5(1), pp.21-33, Mar./Apr. 1992.

[7] J. Jaworski, "Java Developer's Guide", Sams.net, 1996.

[8] P. M. Tyma and G. Torok and T. Downing, "Java Primer Plus", Waite Group, 1996.

[9] S. Brinkkemper and S. Hong and A. Bulth-uis and G. Goor, "Object-Oriented Analysis and Design Methods : a Comparative Review", (c)Univ. of Twente, Jan. 1995.

[10] E. Yourdon, "Java, the Web, and Software Development", Computer, IEEE, Vol.29, No.8, pp.25-30, Aug. 1996.

[11] M. A. Hamilton, "Java and the Shift to Net-Centric Computing", Computer, IEEE, Vol.29, No.8, pp.31-39, Aug. 1996.

[12] The Object Agency Inc., "A Comparison of Object-Oriented Development Methodologies", The Object Agency Inc., 1995.

[13] 최영근, 허계범, "객체지향 소프트웨어 공학", 한국심리학, 1995.

[14] 최성운, 도홍석, 계원경, "객체지향 소프트웨어 개발 방법론 동향", 정보과학회지 제14권 제10호, pp.4-11, Oct. 1996.

[15] 반종오, 최형진, "CASE 웹 서버 구축을 위한 이벤트 다이어그래머의 설계", 한국정보처리학회, 추계 학술발표논문집, pp.526-531, 1996. 10.

[16] 반종오, 최형진, "CASE 서버 구축을 위한 EDA의 프로토타입", 한국정보처리학회, 춘계 학술발표논문집, pp.645-650, 1997. 4.

반 종 오

1994년 강원대학교 전자계산학과(학사)

1997년 강원대학교 전자계산학과(이학석사)

1997년~현재 강원대학교 전자계산학과 박사과정

1998년~현재 한림전문대학 인터넷정보과 전임강사

관심분야 : 인공지능(패턴인식 및 에이전트), S/W공학 등

최 형 진

1982년 영남대학교 물리학과(이학사)

1987년 일본 동경공업대학 정보공학과(공학석사)

1990년 일본 동경공업대학 정보공학과(공학박사)

1990년~91년 한국전자통신연구소 선임연구원

1991년~97년 강원대학교 전자계산학과 조교수

1997년~현재 강원대학교 전자계산학과 부교수

관심분야 : 인공지능, 화상처리, 패턴인식, 컴퓨터 비전, 컴퓨터그래픽, S/W 공학 등