소프트웨어 재사용 시스템을 지원하는 사용자 인터페이스 구축기의 설계 및 구현

김 상 근† 홍 찬 기†† 이 경 환†††

요 약

대부분의 사용자 인터페이스 관리 시스템은 사용자 인터페이스의 대화 모델을 채택하고 있다. 사용자 인터페이스 관리 시스템의 구현은 채택된 사용자 인터페이스의 대화 모델에 의해 강한 영향을 받는다. 모델-뷰-콘트롤러(Model-View-Controller : MVC) 프레임웨이 Smalltalk 환경에서 사용자 인터페이스 개발의 여러 측면에서 기여한 동안 사용자 인터페이스는 강하게 결합된 모델, 뷰, 콘트롤러 클래스를 가진 MVC를 이용하여 생성되었다. 이러한 결합은 소프트웨어 부품의 재사용을 방해하였다. 따라서 본 논문에서는 MVC 모델이 갖는 강한 결합력으로 인한 소프트웨어 재사용의 저하를 해결하기 위해, 상호 작용의 구문적 관리를 하는 다이알로그 객체를 추가하여 사용자로 부터 생성된 메세지가 콘트롤러에 의해 바로 값이 변경되지 않고 다이알로그에 전달되도록 MVCD 모델을 제안하였다. 메세지를 전달받은 다이알로그 객체는 모델 객체의 값을 변경하기 위해 모델 객체를 호출한다. 모델 객체는 active 값들을 가지고 있기 때문에 값의 변경은 연결된 콘트롤러 객체에게 전달된다. 최종적으로 콘트롤러 객체는 새로운 값을 변경하고 뷰 객체를 수정한다. 이러한 사용자 대화 모델에 기반을 둔 사용자 인터페이스 구축기를 X-윈도우상에서 OSF/Motif를 이용하여 개발하였다.

The Design and Implementation of User Interface Builder to support Software Reuse System

Sang Geun Kim,† Chan Ki Hong†† and Kyung Whan Lee†††

ABSTRACT

Most UIMS(User Interface Management System) adopt dialogue model of user interface. Implementation of UIMS influenced by adopted dialogue model of user interface strongly. While the Model-View-Controller framework has contributed to many aspects of user intereface development in Smalltalk environment, user interfaces generated with MVC have highly coupled model, view, and controller classes. Such coupling impedes the reuse of software component. So, In this paper, we suggest MVCD model to resolve a decline of reuse with MVC have highly coupled. User messages are not changed by Controller immediately, but sent to Dialogue object which maintains the syntatic structure of the interaction. Dialogue object invokes Model object to updates its value. Since Model objects have active values, the value change propagates to the linked Controllers. Finally, Controller object convert the new value and update the View object. User interface builder is implemented on X-window with OSF/Motif that is base on this user dialogue model.

1. 서 론

사용자 인터페이스는 사용자와 응용시스템 간의 상호 대화를 위한 대화 창구로 사용자가 응용시스템에 입력을 주고, 응용 프로그램이 사용자에게 결과를 보여준다. 즉, 소프트웨어 공학에서는 사용자와 컴퓨터간의 상호대화를 위한 "공유경계"로 인터페이스가 정의되어 사용되고 있다. 사용자와 컴퓨터간의 상호 대화는 자연스럽게 이루어져야 하며, 사용자에게 편리함을 제공해야 한다. 과거에는 사용자가 컴퓨터에 명령어와 함께 여러가지 옵션들과 인수들을 텍스트로 입력하는 방식의 인터페이스가 사용되었다. 그러나, 이러한 방식은 사용자가 많은 명령어, 인수와 옵션들을 모두 기억하고 있어야 하는 어려움이 있었다. 또한 종래에는 프로그램을 보다 빠르고 효율적으로 작성하는데 중점을 두고 있었으나, 컴퓨

† 정 회 원 : 중앙대학교 컴퓨터공학과 강사
†† 정 회 원 : 관동대학교 전자계산학과 조교수
††† 정 회 원 : 중앙대학교 컴퓨터공학과 교수
논문접수 : 1995년 2월 9일, 심사완료 : 1995년 3월 30일

터의 광범위한 보급과 여러 분야에서의 사용으로 사용자들은 보다 손쉬운 컴퓨터 사용을 요구하게 되었다. 즉, 컴퓨터 조작에 익숙하지 않은 사용자에게는 명령어를 외워서 작업하는 기존의 명령어 지향적인 인터페이스는 사용하기에 어려웠다. 따라서, 사용자가 소프트웨어를 편리하게 사용할 수 있는 인터페이스에 대한 요구가 증대되고 있다[1].

이러한 요구에 부응하여 급속하게 자리잡기 시작한 그래픽 사용자 인터페이스(Graphical User Interface)는 기존의 환경을 탈피하여 보고 느낀 대로 사용자가 작업을 하도록 도와 준다. 한마디로 사용자가 친숙하고 쉽게 접근할 수 있도록 도와 주는 새로운 개념의 도구이다.

소프트웨어를 위한 좋은 사용자 인터페이스를 개발하는 것은 매우 어렵다. 이는 소프트웨어를 사용하기 쉽도록 보장해주는 지침이나 기술이 없고, 소프트웨어 개발자는 자신이 개발한 프로그램이 인간과 같은 능력의 인터페이스를 제공한다는 것을 증명하기 어렵기 때문이다. 또한 인터페이스 소프트웨어는 많은 장치들을 빈번히 제어해야 하고, 비동기적으로 사건의 입력 스트림들을 보내야 하며, 사용자의 행동과 시스템 반응사이에 지체감이 없도록 보장해야 한다는 강력한 요구를 만족해야 하기 때문에 근본적으로 작성이 어렵다[5, 10, 12]. 결과적으로 인터페이스 소프트웨어는 자주 프로토타입으로 만들어지고 반복적으로 수정되어야 한다. 그러므로 사용자 인터페이스 설계와 개발을 위한 도구의 개발에 관한 관심이 집중되고 있다.

따라서 본 논문에서는 재사용 시스템인 CARS [2,15]를 사용하는 사용자가 사용하기 쉬운 사용자 인터페이스를 만들기 위한 객체지향 시각적 사용자 인터페이스 구축기를 설계및 구현하였다. 또한, 이러한 사용자 인터페이스 구축기를 개발하기 위해 필요한 모델로 많은 사용자 인터페이스 구축 시스템에 직접적으로 영향을 주었거나 기반이 되었던 MVC(Model-View-Controller) 모델을 일부 보완한 MVCD(Model-View-Controller-Dialogue) 모델을 제안한다.

이 논문의 구성은 다음과 같다. 2장에서는 기존의 사용자 인터페이스 관리 시스템에서 채택한 사용자 인터페이스 대화 모델과 제안한 MVCD 모델을 비교한후, 기존의 사용자 인터페이스 관리시스템에 대해 살펴보고, 3장에서는 개발하고자 하는 객체지향 사용자 인터페이스 구축기의 구조와 각 기능 정의에 대해 기술하고, 4장에서는 결론에 대해 기술한다.

2. 관련 연구

사용자 인터페이스 소프트웨어는 매우 크고 복잡하여 작성하기 어려울 뿐만 아니라, 수정하기에도 어렵다. 예로 인공지능 응용시스템의 경우를 보면, 전체 원시 코드의 40~50 % 정도가 사용자 인터페이스를 구축하는데 필요한 부분이고, 실행시 메모리는 인터페이스 측면에 집중된다.

따라서, 사용자 인터페이스에 대한 연구가 증가하고 있는 현실이지만, 소프트웨어를 사용하기 쉽도록 보증해주는 지침이나 기술이 없기 때문에 소프트웨어에 대한 좋은 인터페이스를 구축하는 것은 어렵다. 결과적으로 인터페이스 소프트웨어는 자주 프로토타입으로 만들어지고 반복적으로 수정되어야만 한다. 또한, 시스템이 사용하기가 쉬워지면 쉬워질수록, 사용자 인터페이스를 구축하기는 더욱 어려워진다[5]. 이러한 문제점을 해결하기 위해 개발된 사용자 인터페이스 개발 도구를 이용하면, 중요한 두 가지 부분에서 이점을 얻을 수 있다.

첫째는, 프로토타이핑의 결과로 얻는 것으로써, 대부분의 도구들은 소프트웨어 개발에 있어 불필요하게 상세한 것들은 감추고, 빠른 프로토타이핑을 제공하므로 개발 기간을 줄여주므로써 비용 절감을 가져온다. 또한 사용자가 원하는 인터페이스를 제공할 수 있다.

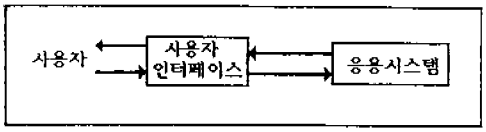
둘째는, 사용자 인터페이스의 인터페이스 코드를 응용 부분과 분리시킴으로써 일관된 사용자 인터페이스를 제공하고, 인터페이스 부분의 수정없이 응용을 수정하도록 함으로써 관련된 응용시스템간의 일관된 인터페이스를 유지하도록 한다. 따라서, 이러한 이점들을 얻기 위해 많은 방법들이 개발되고 시도되었다.

본 장에서는 위와 같은 문제점을 해결하고 이점을 제공하는 사용자 인터페이스 개발 도구와

사용자 인터페이스 개발 도구에서 사용되는 사용자 인터페이스 대화 모델에 대한 살펴보도록 한다.

2.1 사용자 인터페이스 대화 모델 (User Interface Dialogue Model)

사용자 인터페이스 개발 도구에 채택된 모델은 사용자 인터페이스 개발 도구에 의해 생성되는 사용자 인터페이스의 영역을 정의한다. 일반적인 사용자 인터페이스 모델은 (그림 1)과 같다.



(그림 1) 사용자 인터페이스 모델
(Fig.1) User Interface Model

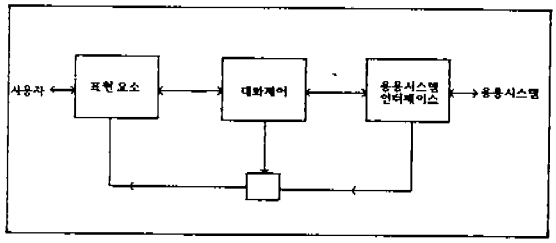
모든 사용자 인터페이스 개발 도구는 사용자 인터페이스의 모델을 채택하고 있다. 이런 사용자 인터페이스 모델은 사용자 인터페이스 개발 도구에서 사용자 인터페이스와 그 구현에 영향을 주는 것을 묘사하기 위해 사용된다. 많은 기술들이 사용자 인터페이스를 기술하기 위해 개발되어 왔다[10].

2.1.1 Seeheim 모델

Seeheim 모델[13, 14]은 사용자 인터페이스에 반드시 나타나야 할 3개의 컴포넌트(표현 요소, 대화제어, 응용시스템 인터페이스)들을 (그림 2)와 같이 보여준다.

표현 요소는 사용자 인터페이스에 대한 외부 표현을 담당하며, 입출력 장치, 스크린 레이아웃, 상호작용 기법(interaction technique), 디스플레이 기법을 포함한다. 사용자 인터페이스의 다른 컴포넌트들은 입출력 장치와 직접 연결될 수 없고, 오직 표현 요소만이 입출력 장치들을 직접 다룰 수 있다.

대화 제어 요소는 사용자와 응용시스템간의 대화 구조를 정의하며, 응용시스템 인터페이스는 사용자 인터페이스와 응용 프로그램간의 인터페이스를 정의한다. 이 컴포넌트는 응용 프로그램의 수행 절차의 호출을 다룬다. 세개의 컴포넌트들은 3개의 논리적으로 분리된 프로세스로 볼 수 있다. 컴포넌트들은 컴파일러에서 사용되는 것과 유사한 토큰을 보냄으로써 대화할 수 있다. 토큰은 이름과 자료값들의 모임으로 이루어져 있다.



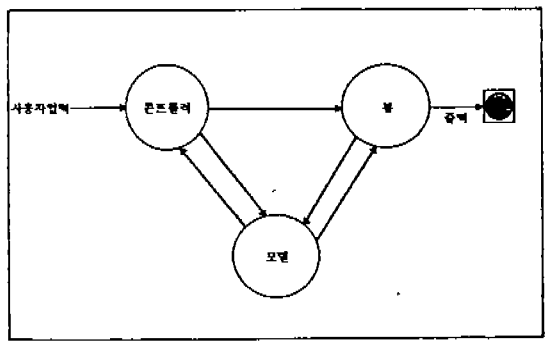
(그림 2) Seeheim 모델
(Fig. 2) Seeheim Model

2.1.2 MVC(Model-View-Controller) 모델

Smalltalk는 실험적인 소프트웨어를 개발하는데 효율적인 환경이다. MVC 모델[3, 4]은 Smalltalk 환경하에서 사용자 인터페이스를 개발하기 위한 중요 수단으로서 개발되었다. MVC 모델은 사용자 인터페이스에 대한 책임을 (그림 3)과 같이 세개의 객체로 나눈다.

모델객체는 응용의 자료구조를 표현하고, 모델의 뷰에 디스플레이되는 정보에 대한 엑세스를 갖고 있다.

뷰 객체는 모든 그래픽 타스크를 다룬다. 자료를 모델에게 요구하고, 받은 자료를 출력한다. 또한 뷰는 서브 뷰(subview)를 가질 수 있고, 수퍼 뷰(superview)에 포함될 수도 있다. 수퍼 뷰는 이러한 서브 뷰/수퍼 뷰 계층 구조간의 그래픽 변



(그림 3) Model-View-Controller 모델
(Fig. 3) Model-View-Controller Model

환, 윈도우 작업을 수행하기 위한 능력을 제공한다.

콘트롤러 객체는 콘트롤러와 관련된 모델, 뷰와 사용자 입력간의 인터페이스를 제공한다. 또한 다른 뷰-콘트롤러 쌍과의 상호작용을 스케쥴링한다.

표준 상호작용 주기는 다음과 같다.

1) 사용자는 임의의 입력 행위를 수행하고, 작동중인 콘트롤러는 모델에게서 적절한 행위를 불러냄으로써 응답한다.
2) 모델은 규정된 오퍼레이션을 수행하고, 모델의 상태를 바꾸고 모델에 종속적인 모든 뷰에 브로드캐스팅한다.
3) 각 뷰는 필요하다면, 뷰의 상태와 디스플레이를 갱신하기 위하여 모델에 질의할 수 있다.

2.2 사용자 인터페이스 개발 도구

사용자 인터페이스 개발 도구는 크게 사용자 인터페이스 툴킷과 사용자 인터페이스 관리 시스템의 두가지 유형으로 분류된다.

2.2.1 사용자 인터페이스 툴킷

대부분의 윈도우 시스템들은 응용 프로그램들이 사용할 수 있는 루틴을 포함한 툴킷을 갖고 있다. 툴킷에는 두가지 종류가 있는데, 하나는 Macintosh의 Toolbox와 같이 응용 프로그램에 의해서 불리워 질 수 있는 절차들의 집합이다. 또 다른 하나는 X-윈도우의 X11 툴킷과 같이 상속을 갖는 객체지향형 프로그램의 형태를 취한다[5].

툴킷 형태는 제한된 상호작용 형태를 제공하고, 종종 사용자 인터페이스를 만들기에 비용이 많이 들고, 사용하기가 어렵다. 전형적으로 툴킷은 많은 상호작용 기법을 구현한 많은 루틴을 포함하므로써, 종종 원하는 인터페이스를 만들기 위해 어떤 절차를 사용해야 하는지가 명확하지 않다.

2.2.2 사용자 인터페이스 관리 시스템

사용자 인터페이스 관리 시스템은 툴킷이 갖는 문제점을 해결하기 위하여 개발되었으며, 사용자와 응용 소프트웨어 간에 디스플레이의 시각 부분과 대화의 모든 측면을 포함하는 인터페이스를 다룬다. 사용자 인터페이스 관리 시스템에는 언어 기반형, 그래픽 명세형, 자동 생성형으로 나눌 수 있다[6].

1) 언어기반형 사용자 인터페이스 관리 시스템

대부분의 사용자 인터페이스 관리 시스템에서 사용자는 특정 목적 언어를 가지고 인터페이스를 명시한다. 상태 천이 다이아그램(Jacob의 STD), context-free 문법(Syngraph), 이벤트 언어(Sassafras), 선언적(declarative) 언어(Open Dialogue), 제한(constraint) 언어(Thinglab), 그리고 객체지향 언어(MacApp)가 여기에 속한다.

2) 그래픽 명세형 사용자 인터페이스 관리 시스템

그래픽 명세형 사용자 인터페이스 관리 시스템을 사용하면 부분적으로 마우스를 이용하여 스크린 상에 있는 객체의 위치를 지정함으로서 인터페이스를 정의할 수 있다. 인터페이스의 시각적 표현이 가장 중요한 요소가 되기 때문에 그래픽 툴을 사용하여 시각적 표현을 정의하는 방법을 사용한다. Cardelli의 DialogEdit와 Myer의 Peridot가 여기에 속한다.

3) 자동 생성형 사용자 인터페이스 관리 시스템

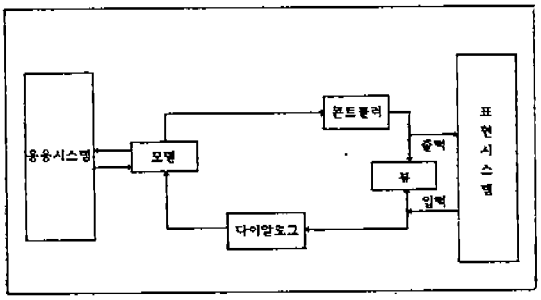
모든 언어기반형 시스템의 문제점은 사용자 인터페이스의 설계와 형식에 대해 많은 부분을 기술해야 한다는 점이다. 따라서 이런 문제점을 해결하기 위하여, 자동 생성형 사용자 인터페이스 관리 시스템은 응용 프로그램의 의미 절차 명세를 가지고 자동적으로 인터페이스를 만들어낸후 설계자에게 인터페이스를 개선하도록 하는 방식을 사용하고 있다. MIKE, ITS, IDL 같은 시스템이 여기에 속한다.

2.3 MVCD(Model-View-Controller-Dialogue) 모델

MVC 모델은 사용자 인터페이스 개발에 있어 많은 영향을 주었다. 많은 사용자 인터페이스 개발 도구가 MVC 모델을 기반으로 하거나 MVC 모델로 부터 영향을 받았다. 그러나 MVC 모델을

가지고 개발된 인터페이스는 모델, 뷰, 콘트롤러 객체가 강하게 결합되었다. 이러한 결합은 소프트웨어 부품의 재사용을 방해하였다.

따라서 본 논문에서는 MVC 모델이 갖는 강한 결합력으로 인한 소프트웨어 재사용의 저하를 해결하기 위해, 상호 작용의 구문적 관리를 하는 다이알로그 객체를 추가하여 사용자로 부터 생성된 메세지가 콘트롤러에 의해 바로 값이 변경되지 않고 다이알로그에 전달되도록 MVCD 모델[16]을 설계하였다. 메세지를 전달받은 다이알로그 객체는 모델 객체의 값을 변경하기 위해 모델 객체를 호출한다. 모델 객체는 active 값들을 가지고 있기 때문에 값의 변경은 연결된 콘트롤러 객체에게 전달된다. 최종적으로 콘트롤러 객체는 새로운 값을 변경하고 뷰 객체를 수정한다.



(그림 4) MVCD(Model-View-Controller) 모델
(Fig. 4) MVCD(Model-View-Controller) Model

1) 모델 객체

Active 값과 함께 응용의 일반적인 자료 구조를 제공한다. 즉, 자동적으로 값의 변화를 전달한다. 사용자 인터페이스 부분과 응용부분간의 명백한 인터페이스를 제공한다. 의미되돌림(semantic feedback)을 지원하기 위해 사용자 인터페이스 부분에 대한 의미 정보를 가져온다.

2) 뷰 객체

모델의 가시적 디스플레이를 나타낸다. 버튼, 메뉴, 스크롤바와 같은 그래픽 객체를 포함한다. 뷰 객체는 일반적으로 그래픽 시스템이 될 수 있는 표현 시스템을 캡슐화한다.

3) 콘트롤러 객체

모델 객체와 뷰 객체간의 인터페이스를 제공한다. 콘트롤러 객체는 정수를 스트링으로 치환하는 것과 같은 데이타 치환을 수행한다. 새로운 모델 객체와 뷰 객체에 대한 요구를 최소화함으로써 높은 유연성을 갖도록 한다.

4) 다이알로그 객체

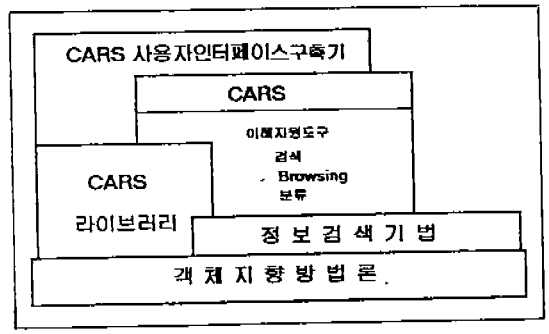
상호작용에 대한 구문 정보를 관리한다. 사용자로부터 생성된 메세지가 콘트롤러에 의해 바로 값이 변경되지 않고 다이알로그에 전달된다. 메세지를 전달받은 다이알로그 객체는 모델 객체의 값을 변경하기 위해 모델 객체를 호출한다. 모델 객체는 active 값들을 가지고 있기 때문에 값의 변경은 연결된 콘트롤러 객체에게 전달된다. 최종적으로 콘트롤러 객체는 새로운 값을 변경하고 뷰 객체를 수정한다.

객체들간의 통신은 링크 개념을 통하여 모델되었다. 링크는 객체들간의 메세지를 전달하는 영구적인 통신 채널이다.

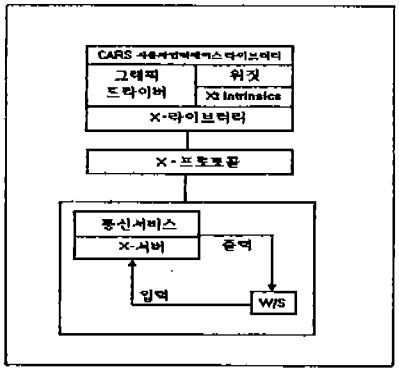
3. 사용자 인터페이스 구축기

재사용 시스템인 CARS를 사용하는 사용자의 노력을 줄여주기 위해 일관된 사용자 인터페이스가 필요하게 되었다. 따라서, 사용자 인터페이스 구축기를 구축하였다.

사용자 인터페이스 구축기는 위짓 기술과 X-윈도우 시스템[8,17]에 기반을 둔 시스템이다. 응용시스템을 개발하는 소프트웨어 개발자가 인터페이스를 부분을 개발하고자 할 때 지원할 수 있다. 사용자 인터페이스를 구성하는 윈도우와 객체들이 라이브러리에 저장되어 추후 이를 사용하려는 다른 응용 프로그램에 의해 재사용되어질 수 있다.



(그림 5) 재사용 시스템 CARS의 구조
(Fig. 5) Architecture of Reuse System CARS



(그림 6) 사용자 인터페이스 구축기의 환경
(Fig. 6) Environment of User Interface Builder

소프트웨어 재사용 시스템인 CARS를 이용하여 응용시스템을 개발하고자 하는 소프트웨어 개발자는 객체지향 시각적 인터페이스 구축기를 이용하여 사용자 인터페이스를 쉽게 개발할 수 있다. 재사용 시스템 CARS의 구조는 (그림 5)와 같다. 따라서 개발한 사용자 인터페이스를 화일로 저장하여 CARS의 라이브러리에 저장하여 두고 재사용할 수 있다. 사용자 인터페이스 구축기의 작업 환경은 (그림 6)과 같다.

3.1 재사용 시스템 CARS

객체지향 환경하에서 재사용 라이브러리와 재사용 시스템인 CARS를 구축하였다. 객체지향 라이브러리의 상속에만 의지하는 제한성을 해결하기 위하여 정보 검색 기법을 적용하였다.

3.2 사용자 인터페이스 라이브러리

라이브러리는 사용자 인터페이스를 구축하는데 필요한 객체들을 제공한다. 라이브러리의 구조는 (그림 7)과 같다.

1) 상위단계 객체들

상위 단계의 객체는 윈도우와 가장 높은 추상화를 이루는 모델 객체및 다른 단계의 객체들에 대한 광역 또는 공통되는 속성을 정의한다. 이러한 객체들은 시스템을 정의하는 객체와 서버를 정의하는 객체, 윈도우를 지정하는 객체, 모델을 설정하는 객체들로 구성된다. 2) 복합 객체들

복합 객체는 메뉴, 스크롤바 등과 같이 기본적인 객체들로부터 생성된 객체들이다. 사용자가 생성한 인터페이스의 부품들도 포함된다.

3) 기본 객체들

기본 객체는 사용자 인터페이스를 구성하는데 필요한 가장 기본적인 위짓과 그래픽 도메인에 필요한 그래픽 객체들로 구성된다. 기본 객체는 사용자 인터페이스의 실제 화면에 나타나게 될 시각적인 구성요소를 의미한다.

4) 하위단계 객체들

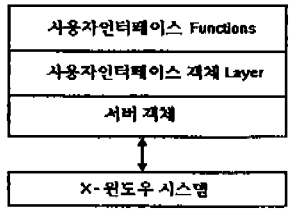
객체의 속성을 규제하기 위한 객체들로 사용자들에게는 감추어진 객체이다. 또한 다형성(Polymorphism)을 통하여 다른 객체들간의 공통연산의 인터페이스를 지원한다.

5) 사용자 인터페이스 Functions

사용자 인터페이스 함수들은 객체지향 방법을 통하여 개발되었고, 대화적이고 유용한 정보를 전달함으로써 사용자 인터페이스 개발을 지원한다.

3.3 객체 기술자

사용자 인터페이스 구축기의 사용자에 의해 사용되는 객체의 정보를 지식으로 표현하는 방법으



(그림 7) 라이브러리의 구조
(Fig. 7) Library Architecture

```
#description
  #class class_name
    #object object_name
    {object _ name.attribute _ name : attribute _
    value }
    #end object
  #end class
#end descriotion
```

(그림 8) 객체 기술자의 기본 형식
(Fig. 8) Basic form of Object descriptor

로 객체 기술자(Object Descriptor)를 제시한다. 객체 기술자는 객체의 표현에 필요한 추출 정보를 입력으로 객체의 표현 화일을 생성하는 기본이 된다. 객체 기술자는 정보의 구조와 각 요소의 특징을 표현할 수 있도록 하였다. 객체 기술자의 기본 형식은 (그림 8)과 같다.

이를 BNF(Backus-Naur Form) 형태로 표현하면 (그림 9)와 같다.

객체 기술자로 pushbutton을 표현 한 예는 (그림 10)과 같다.

```
#description
  #class XmPushButton
    #object pushButton
      pushButton.labelString:OK
      pushButton.labelType:XmString
      pushButton.mappedWhenManaged:True
      pushButton.Height:20
      pushButton.Width:40
      pushButton.X:60
      pushButton.Y:120
      .......
    #end object
  #end class
#end description
```

(그림 9) 객체 기술자의 BNF 기술 형식
(Fig. 9) BNF of Object Descriptor

```
⟨Obj_Des⟩::=⟨DES⟩ ⟨O_Des⟩
⟨DES⟩::=#description
⟨O_Des⟩::=⟨CLSS⟩ ⟨END_DES⟩
⟨CLSS⟩::=⟨CLS⟩ ⟨CLSS_body⟩
⟨CLS⟩::=#class ⟨NAME⟩
⟨CLSS_body⟩::=⟨OBJ⟩ ⟨END_CLSS⟩
⟨OBJ⟩::=⟨OBJ_body⟩ ⟨END_OBJ⟩
⟨OBJ_body⟩::=⟨OBJ_NA⟩ ⟨ATTRBT⟩
⟨OBJ_NA⟩::=#object ⟨NAME⟩
⟨ATTRBT⟩:: = {⟨ATTRBT _ NAME⟩:
                ⟨ATTRBT_VALUE⟩}
⟨END_OBJ⟩::=#end object
⟨END_CLSS⟩::=#end class
⟨END_DES⟩::=#end description
⟨NAME⟩::=⟨NAME⟩ | a | b |  | y | z | A |
          B |  | Y | Z | Null
⟨ATTRBT_NAME⟩::=⟨NAME⟩
⟨ATTRBT_VALUE⟩::=⟨NAME⟩ | 0 | 1 |
                 | 8 | 9 | Null
```

(그림 10) pushButton의 예
(Fig. 10) Example of pushButton

3.4 사용자 인터페이스 구축기

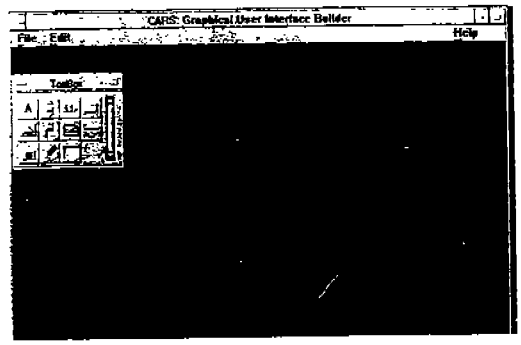
사용자 인터페이스 구축기는 X-윈도우/Motif로 RS/6000에서 개발되었으며, 3개의 독립적 윈도우인 툴박스(tool Box) 윈도우, 브라우저(browser) 윈도우, 자원 편집기 윈도우로 구성된다. 사용자 인터페이스 구축기의 초기 화면은 (그림 11)과 같다.

사용자 인터페이스 구축기가 제공하는 명령어들은 다음과 같다.

1) File 명령어

File 명령어는 화일과 관련되어 수행되는 다음과 같은 여러개의 서브 메뉴를 가지고 있다.

- Open : 사용자가 이미 라이브러리에 존재하는 인터페이스 화일들중에서 원하는 화일을 열 수 있다. 마우스를 원하는 화일에 위치시키고 마우스를 두번 클릭함으로써 그 화일을 선택할 수 있다.
- New : 이 명령어를 이용하여 사용자가 새로운 인터페이스 화일을 생성할 수 있다.
- Save : 현재 작업중인 사용자 인터페이스 화일을 저장한다. 만일 인터페이스 데이타가



(그림 11) 사용자 인터페이스 구축기의 초기화면
(Fig. 11) Initial Screen of User Interface Builder

이미 저장되어 있다면, 현재 작업중인 화일이 존재하는 디렉토리에 단순히 overwrite한다.

- Save As : 현재 작업중인 화일을 다른 이름으로 저장한다.
- Save All : 현재 열려 있는 모든 인터페이스 화일을 저장한다.
- Close : 현재 작업중인 화일을 닫는다.

2) Edit 명령어

Edit 명령어는 표준 편집 명령어를 포함하는 서브 메뉴를 가지고 있다.

- Cut : 사용자가 작업중인 내용 일부를 다른 곳으로 옮기고자 할 경우, 선택된 부분을 작업 영역에서 삭제하고, 다른 곳으로 복사 또는 이동시키기 위하여 클립보드에 저장하여 둔다.
- Copy : 사용자가 원하는 내용을 복사하는 기능이다. 사용자가 어떤 내용을 복사하여 다른 곳에 그 내용을 복사하려면, 복사하고자 선택한 부분을 클립보드에 저장한다.
- Paste : 사용자가 복사하고자 클립보드안에 저장중인 내용을 원하는 곳에 삽입한다.
- Delete : 사용자가 선택한 부분을 클립보드에 저장하지 않고 삭제한다.

3) Help 명령어

Help 명령어는 사용자가 사용자 인터페이스 구축기에서 제공하는 기능의 상세한 사용법등을 제공한다.

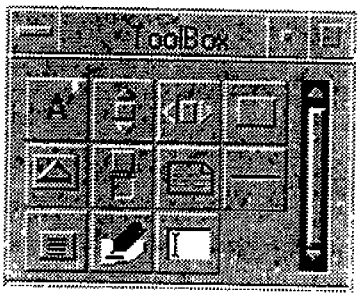
3.2.1 툴박스 윈도우

사용자 인터페이스 구축기의 사용자는 툴박스 윈도우안에 있는 위짓 아이콘들중 사용자 인터페이스를 구축할 때 필요한 아이콘을 마우스로 클릭하여 원하는 위치에 놓으면 선택된 그래픽 사용자 인터페이스의 구성 요소를 생성할 수 있다.

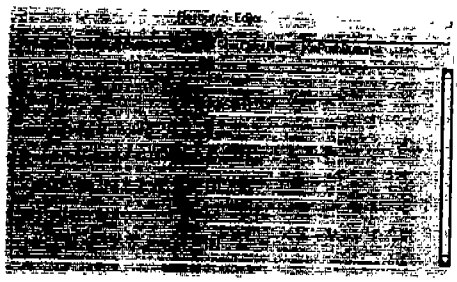
사용자가 look-and-feel을 지원하는 WYSIWYG (What You See Is What You Get) 기능이 기본적으로 제공된다. 또한 툴박스에 있는 위짓을 이용하여 유용한 위짓을 만들고, 이를 라이브러리에 저장하여 추후에 재사용할 수 있다. 툴박스 윈도우는 (그림 12)와 같다.

3.2.2 자원 편집기(Resource Editor) 윈도우

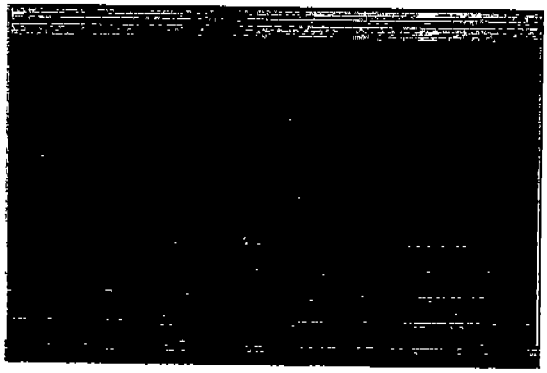
그래픽 사용자 인터페이스를 구성하는 모든 위짓은 자원 편집기를 이용하여 편집, 수정할 수 있다. 또한, 하나의 위짓이 가지는 자원을 수정하게 되면 즉시 그래픽 사용자 인터페이스 구성 프로그램의 형태는 영향을 받아, 화면상에 변경된 모습으로 다시 표현된다. 자원 편집기의 형태는 (그림 13)과 같다.



(그림 12) 툴박스
(Fig. 12) Toolbox



(그림 13) 자원 편집기
(Fig. 13) Resource Editor



(그림 14) 사용자 인터페이스 브라우저
(Fig. 14) User Interface Browser

3.2.3 브라우저 윈도우

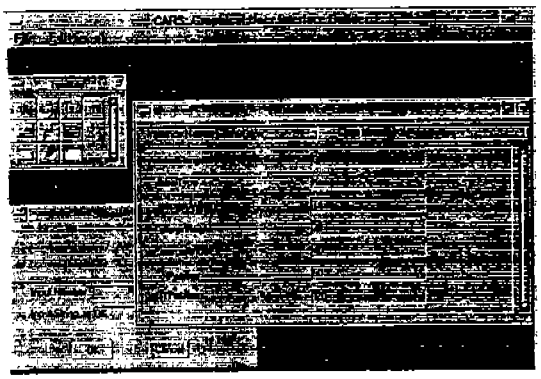
브라우저 윈도우는 사용자가 선택하여 구성한 그래픽 사용자 인터페이스를 Motif의 위짓 계층 구조로 나타내 준다. 이 브라우저는 재사용 시스템인 CARS의 검색시스템에서 제공하는 브라우저를 기반으로 구축하였다. 따라서, 사용자는 브라우저 윈도우의 panner를 이용하여 빠르고 쉽게 위짓의 계층 구조를 한눈에 파악할 수 있다. 브라우저는 (그림 14)와 같다.

3.3 실행 예

이 절에서는 사용자가 push 버튼을 생성한후 해당 자원을 편집하는 예를 보이도록 하겠다. 사용자는 라이브러리에서 topLevelShell을 선택하여 작업 영역을 만들고, 그 위에 form과 RowColum을 선택한다.

그리고 툴박스를 이용하여 원하는 라벨 위짓을 선택하면 Label1이라는 기본값을 가지고 디스플레이된다. 이렇게 생성된 Label1이라는 기본 값을 가진 라벨 위짓을 클릭하여 선택하고, 복사 명령을 선택하여 Label2라는 새로운 라벨을 생성한다. 이렇게 생성된 라벨 위짓들을 자원 편집기를 이용하여, 이름을 "Input Number"와 "Input String"으로 각각 변경한다. 푸시버튼도 같은 방법으로 생성한다. 선택된 푸시버튼은 pushButton1이라는 기본 label을 가지고 작업 영역에 디스플레이 된다. 이렇게 생성된 푸시버튼을 클릭하여 선택하고 복사 명령을 선택한 후 pushButton2라는 label을 가진 새로운 푸시버튼을 생성한다. 이렇게 생성된 푸시버튼들의 label을 자원 편집기를 이용하여, 속성을 바꾼다. 자원 편집기를 통하여 변경된 속성 값들은 사용자 인터페이스로 반영된다.

실행 예는 (그림 15)와 같다.



(그림 15) 사용자 인터페이스 구축기의 실행 예
(Fig. 15) Execution Example

4. 평 가

사용자 인터페이스에 대한 평가는 매우 어렵고 아직까지 정확한 기준이 확립되어 있지 않다[5]. 사용자 인터페이스는 가능하면 사용자가 사용하기 쉬워야 하며, 사용자에게 적절한 의미를 되돌려 주어야 하며, 확장성과 일관성이 있어야 한다.

본 논문에서는 사용자 인터페이스 구축기를 다음의 요소에 의해 평가한다.

첫째, 사용의 용이성이다. 사용자 인터페이스 구축기를 이용하여 사용자 인터페이스를 구축하기에 어려움이 없어야 한다. 본 시스템은 point-and-click 방식으로 사용자가 눈으로 보는 것을 그대로 느끼며 작업을 할 수 있는 방식을 제공함으로써 사용의 편리성을 도모하였다.

둘째, 사용자에 대한 의미 되돌림이다. 현재 상태의 내용을 사용자에게 알려주는 의미 되돌림은 사용자와의 인터페이스에 있어서 매우 중요한 요소이다. 객체가 사용자로부터 선택이 되었는지 아닌지를 사용자가 구분할 수 있도록 선택된 객체들은 반전 상태가 되도록 하였다.

셋째, 시스템의 확장성이다. 사용자가 새롭게 작성한 사용자 인터페이스를 라이브러리에 새롭게 등록하여 시스템을 확장할 수 있다. 이와 같이 사용자들은 시스템에서 제공하는 프리미티브 객체이외에 자신이 원하는 객체들을 정의하여 라이브러리에 등록할 수 있다.

넷째, 시스템의 재사용성이다. 사용자가 라이브러리에 등록한 객체들을 사용자들이 원할때마다 사용함으로써, 사용자들의 작업시간을 단축시켜 줄뿐 아니라, 일관성있는 작업을 유도하여 품질 향상의 기대를 가져올 수 있다.

5. 결 론

대부분의 소프트웨어 개발자들은 빠르고 효과

적인 소프트웨어를 좋은 인터페이스와 함께 제공하기를 원한다. 이러한 요구에 따라 사용자 인터페이스 관리 시스템은 많은 발전을 해왔다. 그러나, 소프트웨어를 사용하기 쉽도록 보증해주는 지침이나 기술이 없기 때문에 소프트웨어에 대한 좋은 인터페이스를 구축하는 것은 어렵다. 결과적으로 인터페이스 소프트웨어는 자주 프로토타입으로 만들어지고 반복적으로 수정되어야만 한다. 따라서 많은 소프트웨어 개발자들이 사용자 인터페이스 구축기에 대해 연구하고 있다.

본 논문에서는 사용자 인터페이스를 구축하는 것을 지원하는 사용자 인터페이스 구축기를 설계하고 구현하였다. 또한 사용자 인터페이스 구축기를 개발하는데 필요한 대화 모델(Dialogue Model)로 기존의 MVC 모델을 일부 보완한 MVCD (Model-View-Controller-Dialogue) 모델을 제안하고, 사용자 인터페이스 구축기를 X 윈도우 상에서 OSF/Motif를 이용하여 개발하였다.

사용자 인터페이스 구축기는 툴박스, 브라우저, 자원 편집기를 제공한다. 이러한 환경하에서 point-and-click 방식의 대화형 방식을 지원함으로써 사용자의 이용의 편리성을 도모하였다. 또한, 현재 상태의 내용을 사용자에게 알려주는 의미 되돌림을 고려함으로써 객체가 사용자로부터 선택이 되었는지 아닌지를 사용자가 구분할 수 있도록 하였다.

사용자가 새롭게 작성한 사용자 인터페이스를 라이브러리에 새롭게 등록하여 시스템을 확장할 수 있고, 사용자가 라이브러리에 등록한 객체들을 사용자들이 원할때마다 사용함으로써, 사용자들의 작업시간을 단축시켜줄뿐 아니라, 일관성있는 작업을 유도하여 품질 향상의 기대를 가져올 수 있다. 또한 본 시스템은 사용자 인터페이스와 응용 부분을 분리함으로써 응용이 바뀌어도 인터페이스는 변하지 않는 일관된 사용자 인터페이스를 제공해줄 수 있고, 프로토타이핑을 지원하므로써 빠른 개발을 유도하고 전체 개발 비용을 절감할 수 있다. 아울러 소프트웨어 재사용 시스템의 라이브러리에 개발한 사용자 인터페이스를 저장하여 두고, 재사용 시스템의 사용자가 응용 시스템을 개발하고자 할때 사용자 인터페이스를 쉽게 구축할 수 있다.

참 고 문 헌

[1] 권기현, 시각 질의를 위한 아이코닉 인터페이스에 관한 연구, 중앙대학교 박사학위논문, 1990.

[2] 이경환외, 소프트웨어 재이용 시스템 개발(II), 연구보고서, 과학기술처, 1993.

[3] A. Goldberg and D. Roboson, Smalltalk-80: the Language and Its Implementation, Addison-Wesley, 1983.

[4] A. Goldberg, Smalltalk-80:The Interactive Programming Environment, Addison-Wesley, 1983.

[5] B. A. Myers, "User-Interface Tools:Introduction and Survey," IEEE Software, pp. 15-23, Jan. 1989.

[6] B. Myers, "State of the Art in User Interface Software Tools," Technical Report CMU-CS-92-114, 1992.

[7] B. Schneiderman, "Direct Manipulation:A Step Beyond Programming Languages," IEEE Computer, pp. 57-69, Augest 1983.

[8] D. A. Young, The X WIndow System Programming and Applications with Xt OSF/Motif Edition, Prentice-Hall, 1990.

[9] D. G. Bobrow, S. Mittal, and M. J.Stefik, "Expert Systems:Perils and Promise," Communication ACM, pp. 880-894, Sept. 1986.

[10] D. R. Olsen, "ACM SIGGraph Workshop on Software Tools for User-Interface Management," Computer Graphics, pp. 71-147, April 1987.

[11] G. E. Krasner and S. T. Hope, "A Cookbook for using the Model-View-Controller User Interface Paradigm in Smalltalk-80," Journal of Object-oriented Programming, 1 (3), Auguest/September, pp. 26-49, 1988.

[12] H. Rex Hartson and Deborah Hix, "Human-Computer Interface Development: Concepts and Subsystems for its Management," ACM Computing Surveys, 21(1), March 1989.

[13] Mark Green, "A Survey of Three Dialogue Models," ACM Transaction on Graphics, Vol. 5, No. 3, pp. 244-275, July 1986.

[14] R. Kazman, L. Bass, G. Abowd, and M. Webb, "Analyzing the Properties of User Interface Software," Technical Report CMU-CS-93-201, 1993.

[15] S. G. Kim, et al, "CARS:Computer Aided Reuse System," Conference Proceedings of JCSE'92, pp. 25-32, 1992.

[16] S. G. Kim, et al, "The GUI Builder to support Software Reuse System," Conference Proceedings of JCSE'93, pp. 77-81, 1993.

[17] S. Kobara, Visual Design with OSF/Motif, Addison-Wesley Publishing Company, Inc., 1991.

김 상 근

1987년 중앙대학교 공과대학 전자계산학과 졸업(이학사)
1989년 중앙대학교 대학원 소프트웨어공학 전공(이학 석사)
1993년 중앙대학교 대학원 박사과정수료(소프트웨어공학 전공)
1993년~현재 중앙대학교 컴퓨터공학과 강사
관심분야 : 소프트웨어공학, 사용자 인터페이스, 객체지향 모델링.

홍 찬 기

1986년 중앙대학교 공과대학 전자계산학과 졸업(이학사)
1988년 중앙대학교 대학원 소프트웨어공학 전공(이학 석사)
1992년 중앙대학교 대학원 소프트웨어공학 전공(공학 박사)
1992년~현재 관동대학교 전자계산학과 조교수
관심분야 : 소프트웨어공학, 사용자 인터페이스, 객체지향 시스템.

이 경 환

1980년 중앙대학교 대학원 응용수학 전공(이학박사)
1982년~1983년 미국 Aurban 대학 객원교수
1986년 서독 Bonn 대학 객원교수
1971년~현재 중앙대학교 컴퓨터공학과 교수
관심분야 : 소프트웨어 공학, 객체지향 모델링, 소프트웨어 재사용.