論文95-32B-1-18

구조적 기술에 의한 전문가 시스템의 사용자 인터페이스 개발 방법

(A Development Method of User Interface Using a Structural Description Schema in an Expert System)

金 相 吉 *, 金 成 勳 **, 朴 忠 植 ***, 金 在 熹 **

(Sang-gil Kim, Seong Hoon Kim, Choong-shik Park, and Jaihie Kim)

요 약

본 논문에서는 전문가 시스템의 사용자 인터페이스 모듈을 구현하기 위하여 사용자 인터페이스의 구조적 스키마(description schema)로서 수트(SUIT: Schematic User Interface Tasks)를 정의하였으며, 이를 관리하는 '수트관리기'(SUIMAN: SUIT Manager)를 구현하였다. 수트는 전문가 시스템의 사용자 인터페이스 설계 초기 단계에서, 수트를 사용하여 사용자 인터페이스 구현을 위해 이루어져야 할 행위를 개념적인 상위 수준에서 기술한다. 수트관리기는 수트에서 기술된 것이 언제 실행될지 결정한다. 그래픽 사용자 인터페이스의 실현은 X-윈도우의 MOTIF 라이브러리에 의해 제공되는 함수들을 실행함으로써 이루어진다. 즉, 수트관리기는 수트에 기술된 사용자 인터페이스 명세를 해석하여 실제 그래픽 사용자 인터페이스를 자동으로 실현시킨다. 수트와 수트관리기를 사용함으로써 전문가 시스템 개발에서 사용자 인터페이스 모듈의 보다 신속한 개발이 가능하다. 본 논문의 사용자 인터페이스 기술 방법은 '자동 회선 구성 전문가 시스템'의 사용자 인터페이스에 사용되었다.

Abstract

In this paper, we define Schematic User Interface Task(SUIT) as a scheme which is a conceptual unit to specify the dialogue behavior between man and machine. Using a set of SUITs, the user interface can be described separately from domain applications and can be realized through SUIT Manager(SUITMAN), an execution module devised to interprete and process the descriptions of SUITs. SUIT makes it possible to describe conceptual behaviors performed on the interactions of user interface in early stages of expert system development. SUITMAN analyzes the specification described in SUITs and automatically implements the user interface by using the functions in MOTIF library of X-Window system. By an example of SUIT and SUITMAN to the user interface, we applied them into an expert system, 'Circuit Provisioning Expert System'.

* 準會員, ** 正會員, 延世大學校 電子工學科
(Dept. of Elec. Eng., Yonsei Univ.)
*** 正會員, 永同工科大學 電子計算學科
(Dept. of Computer Science, Yongdong Institute of Tech.)
接受日字 : 1994年 5月 25日

Ⅰ. 서 론

최근 사용자에게 친숙한 응용 소프트웨어를 개발하기 위해 사용자 인터페이스의 프로그램 양이 상당히 증가하였으며, 전문가 시스템 개발에 있어서도 심각한 문제로 대두됨에 따라 사용자 인터페이스의 빠른 개발과 저하된 응용 프로그램 개발의 생산성 향상을 위한 연구가 중요한 주제로 부각되고 있다.[1][2][3] 프로그래밍 노력의 감소를 통한 생산성 향상과 양질의 사용자 인터페이스 구현을 목표로 한 연구는 크게 세가지 방향-사용자 인터페이스 도구 세트, 사용자 인터페이스 관리 시스템(User Interface Management System: UIMS) 및 설계 도구(Design Tool)-으로 나뉘어진다.[4]

사용자 인터페이스 도구 세트는 사용자 인터페이스 설계를 위한 프로그래밍 모듈을 라이브러리형식으로 제공하고 운영 시스템의 표준 언어를 통하여 그래픽 객체를 호출하여 사용한다. 이는 프로그램 개발 노력을 많이 줄였으나, 전체 시스템이 문제 분야 관련 부분과 밀접히 연결된 구조를 가지게 되고, 프로그래머는 이를 사용하기 위해 많은 프로그래밍 지식을 습득해야 한다.

사용자 인터페이스 관리 시스템(UIMS)은 사용자 인터페이스 부분과 문제 관련분야 모듈과의 완전한 분리를 목표로 하여 1982년 David Kasik에 의해 처음 제시되었다[5][6]. 이는 추상적인 상위 레벨의 특별한 언어를 제공함으로써 프로그래머의 생산성 향상을 이룩하였으나, 인터페이스 모듈과 문제 분야 관련 모듈과의 직접적인 통신을 제한하기 때문에, 신속한 인터페이스를 이룩하기 어렵다.

설계 도구는 라이브러리와 직접 조작 방식(Direct Manipulation)을 통해 사용자 인터페이스를 구현하며 프로그래밍의 직접적인 코딩을 피하려는 시도이다. 이것은 UIMS의 사용자 인터페이스 모듈 생성을 용이하게 한 것으로 UIMS의 일종으로 볼 수 있다.

하지만 이러한 개발 도구를 사용하여 시스템을 구축하여도, 전문가 시스템에서는 사용자 인터페이스 모듈과 문제 분야 관련 모듈과의 분리는 어렵다. 왜냐하면, 전문가 시스템에서는 사용자 인터페이스를 구축하기 위하여 심볼 레벨의 지식 표현에서 직접 사용자 인터페이스의 모듈을 구동시켜야 하기 때문이다. 즉, 사용자 인터페이스의 제어 흐름을 문제분야(Domain) 관련 모듈에서 관리하므로 두 모듈이 밀접히 연결된 형태가 된다. 심볼 레벨의 지식 표현은 전체 구조를 파악하기 어렵기 때문에 전체 시스템의 제어 흐름을 한 눈에 파악하기 어렵다.

이러한 문제를 해결하기 위해서 문제 영역(Problem Domain), 사용자 인터페이스 영역(User Interface) 및 제어 흐름(Control Flow)을 각각 분리하여 독립적으로 기술할 수 있는 정형적인 틀이 필요하다. 시스템 개발자가 통일된 형식에 따라 사용자에게 보여지는 모양 또는 형식을 기술할 뿐만 아니라, 기술된 각 부분들 간의 관계를 정형적으로 기술한다. 정형적인 틀을 객체로 설정하고, 이 객체를 다이어그램으로 보여줌으로써 전체 시스템의 제어 흐름을 일목요연하게 파악할 수 있도록 해야 한다.[12]

이 논문에서는 사용자 인터페이스의 개념적 기술 단위인 수트(SUIT: Schematic User Interface Tasks)[7]를 정의하여, 전문가 시스템 문제 분야의 지식 설계와 병행하여 사용자 인터페이스 및 시스템의 제어 흐름을 기술할 수 있도록 하였다. 수트는 Seeheim 모델[8]에서 제안된 것처럼 '보여줄 대상'과 '보여주는 방법'을 분리하여 각각 독립적으로 설계할 수 있도록 하였다. 수트는 사용자 인터페이스 기술에 필요한 요소[9]를 객체로 나타내고, 객체 모델링[10][12]에서 요구되는 조건을 만족하도록 사용자 인터페이스 구현 환경에 무관하도록 정의되었다. 또한, 기술된 수트 및 이들간의 관계를 그림으로 보여주는 수트 다이어그램을 정의하여 전체 시스템 제어흐름을 한 눈에 파악할 수 있다. 수트에 기술된 내용을 처리하고, 실제로 사용자인터페이스를 실현시키는 작업은 수트관리기에서 처리된다. 수트관리기는 X윈도우의 Motif 라이브러리와 Nexpert Object ver. 2.0B를 이용하여 구현되었다.

Ⅱ. 수트(SUIT: Schematic User Interface Task)

1. 사용자 인터페이스 기술의 구성요소

시하임 모델은 사용자와 응용 프로그램 사이에 인터페이스를 중계하는 부분을 세 요소로 나누어 응용 프로그램과 사용자 인터페이스 모듈의 독립성을 보장하고 있다[11]. 전문가 시스템에 적합한 사용자 인터페이스를 쉽게 제작하기 위해서는, 문제 분야의 지식과 사용자 인터페이스 모듈을 분리하여 구축할 수 있도록 시하임 모델의 각 요소가 갖는 정보를 나타낼 수 있는 방법이 고안되어야 할 뿐만 아니라, 추론 과정 중 사용자에게 의미있는 피드백을 주기 위해서는 지식 레벨 또는 상위 레벨에서 사용자 인터페이스를 기술할 수 있는 방법이 필요하다. 이를 위해서는 지식 레벨에서 사용자 인터페이스를 위한 작업과 내용을 추상화하여 기술할 수 있는 객체가 정의되어야 한다. 즉, 그림 1에

서와 같이 사용자에게 직접 보여지는 객체의 형태에 대한 정보를 갖는 표현 부분, 문제 분야의 지식과 문제 분야 전문가의 문제 풀이 과정을 나타내는 추상화 부분, 그리고 표현 부분과 추상화 부분간의 관계 기술하는 변환 제어 부분이 정의되고 기술되어야 한다.

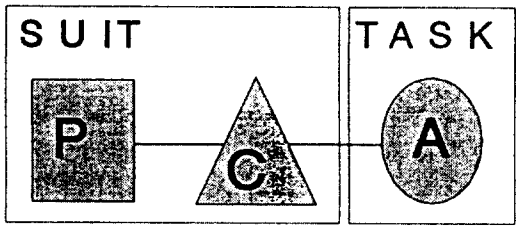


그림 1. 사용자 인터페이스의 기술을 위한 필요 요소

Fig. 1. Three Elements for User Interface Description.

1) 표현부분(Presentation)

표현부분은 스크린 상에서 조작자가 직접 볼 수 있는 객체로서 응용 프로그램내의 지식 덩어리 또는 태스크 객체 그리고 데이타 객체에 대한 외양이다. 이는 '문제 분야의 각 요소들을 사용자에게 어떻게 보여 줄 것인가'에 대한 정보를 담고 있는 부분이다. 모든 정보는 데이타로서 기술된다. 이는 사용자 인터페이스를 구축하기 위해 사용되는 도구에 관련된 부분이고, 이를 실제 실현하기 위해서는 필요한 실행 모듈을 수행시켜야 한다.

2) 추상화 부분(Abstraction)

추상화 부분은 문제분야에 관련된 일을 처리하는 객체로서, 저자가 이미 발표한 바 있는 문제분야의 태스크 개념을 이용하여 기술된다[12]. 전문가 시스템에서의 추상화 부분은 문제 해결을 위해 전문가가 수행하는 작업을 분석/추출하고 이를 추상화하여 여러 계층을 가지는 객체로 구현된다. 각 객체는 여러 하부 작업으로 나뉘어질 수 있고, 자신의 작업을 수행하기 위한 지식과 자료를 가진다.

3) 변환 제어 부분(Transformation Control)

표현 부분과 추상화 부분의 연결성을 기술하는 객체이다. 이는 추론 도중에 추상화 부분의 상태에 따라 사용자에게 보여지는 외양적인 측면, 즉 표현 부분에 어떻게 관련되는지에 관한 정보를 갖는다. 예를 들어, 추론도중에 데이타 입력이나 객체 상태 표시와 같은 사용자와의 인터페이스가 필요할 경우, 추상화 부분에 외양을 기술한 표현 부분을 연결한다. 또한 사용자 인터페이스 실행중에 추론 명령의 실행이 필요할 경우, 표현 부분에 추상화 부분을 연결한다. 연결된 표현 부분이 추론 과정 중 추상화 부분의 상태 변화에 따라 자동으로 표현 부분의 모습이 바뀌는 것도 변환 제어 부분에서 담당한다.

2. 수트(SUIT: Schematic User Interface Tasks)의 정의

수트는 사용자 인터페이스를 분리하기 위해 '태스크 레벨'[12]에서 정의된 사용자 인터페이스 사양(User Interface Specification)의 개념적인 기술 단위'이다. 이것은 사용자 인터페이스를 기술하기 위한 요소중 표현과 변환 제어에 관한 정보를 갖는 객체이다. 즉, 사용자에게 보여지는 객체의 외양과 태스크의 상태에 따른 객체의 외양 변화를 기술한 자료로 구성된다.

표 1. 수트의 기술항목

Table 1. Property List of SUITs.

객체	기술 항목	의 미
수트	name	수트 객체의 이름
	P_Obj_name	표현 객체의 이름
	C_Obj_name	제어 객체의 이름
	Up_SUIT	부모 수트 객체
	Sub_SUIT	부수트 객체
	Sub_SUIT_Type	부수트 처리 형태
	state	수트의 상태
	data test	실행되기 위한 조건
표현 객체	name	표현 객체의 이름
	SUIT_name	자신이 속한 수트의 이름
	adjusted_flag	표현의 변화여부
	realized_flag	표현의 실현여부
제어 객체	name	제어 객체의 이름
	SUIT_name	자신이 속한 수트의 이름
	Linked_Obj_name	의존된 객체의 이름
	Linked_Obj_type	의존된 객체의 형태
	Direction	의존 방향
	Linked_Obj_state	의존된 객체의 상태
	Linked_Obj_newstate	의존된 객체의 변화된 상태
	Focus_SUIT	초점 수트

표 1에서 보인 바와 같이, 수트는 몇 가지 속성(attributes)을 가진다. 수트간의 계층적인 구조를 나타내기 위하여 속성 'Up_SUIT'와 'Sub SUIT'를 갖고 있다. 이 계층구조는 각 객체 간의 종속성을 규정한다. 하나의 수트는 여러 개의 부수트(SubSUIT)로 나뉘어진다. 수트는 부수트를 어떻게 처리할 것인가에 대한 정보(Sub_SUIT_Type)와 시스템의 수행 상황에 따른 각 수트의 상태(state)를 담는 속성을 가진다.

하나의 수트는 구성요소인 표현 객체(presentation object)와 제어 객체(control object)를 부객체(subobject)로 갖는다.

1) 표현 객체의 정의

표현 객체는 사용자에게 보여지는 모습을 규정하는

부분이다. 이는 사용자 인터페이스를 기술하기 위한 요소중 표현에 관한 정보를 갖는다.

표현 객체는 자신과 수트의 연결 관계를 나타내기 위해 'SUIT_name'이라는 속성을 갖고 외양의 변화를 'adjusted_flag'와 'realized_flag'라는 속성으로 나타낸다. 이 플래그는 수트의 상태를 결정하는데 사용된다. 표현 객체는 객체가 나타내는 것에 따라 윈도우(Windows) 또는 표현(Presentation) 클래스에 속하여 윈도우의 크기나 위치 등에 관한 속성을 전수받아 특정 표현 객체의 모양을 기술하기 위한 속성들을 확보한다.

2) 제어 객체의 정의

제어 객체는 사용자 인터페이스를 기술하기 위한 요소중 변환 제어 부분에 해당하며, 수트와 태스크와의 관계와 수트들간의 관계를 규정한다. 제어 객체에 기술된 내용에 따라, 추론 과정에서 수트의 상태가 어떻게 변화되어야 하는지를 결정한다.

제어 객체는 수트와 연결된 태스크 또는 수트 객체에 관련된 정보를 갖는 속성들, 'Linked_Obj_name', 'Linked_Obj_type', 'Direction'을 갖는다. 표현 부분에 태스크를 연결한 경우, 속성 'Linked_Obj_name'은 태스크의 이름을 갖고, 속성 'Linked_Obj_type'과 'Direction'에는 각각 'TASK'와 'in' 또는 'out'을 갖는다. 수트에 연결된 경우는 이들은 각각 수트의 이름과 'SUIT', 그리고 'in' 또는 'out'가 된다. 또한, 추론 과정에서 연결된 객체의 상태 변화에 관한 정보를 갖는 속성(Linked_Obj_state, Linked_Obj_newstate)을 가진다. 시스템 수행 중 이 속성 값에 따라 표현 객체의 모습이 결정된다. 현재 사용자의 이벤트를 받을 수 있는지를 알리는 속성(Focus_SUIT)을 갖는다.

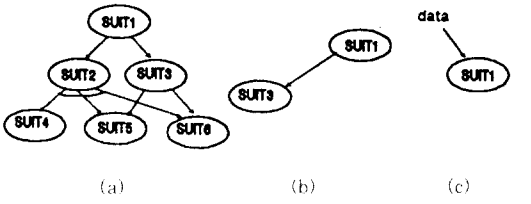
3. 수트 다이어그램

수트에 의하여 기술되는 사용자 인터페이스는 수트 다이어그램에 의하여 표현이 가능하다. 다이어그램을 구성하는 기본 요소를 그림 2에서 보였다. 전체 시스템의 사용자 인터페이스는 수트가 모여서 구성되는데, 수트는 전문가 시스템 제작 과정 중 사용자 인터페이스를 설계하여 시스템을 구성하는 관점에서 3가지의 측면, 즉 구조적 측면, 동적 측면 및 기능적 측면으로 관찰된다. 시스템 구성 측면에서 본 수트의 3가지 측면은 객체지향 기법의 3가지 모델링에 대응시킨 것이므로, 객체 지향 기법의 다이어그램을 변형함으로써 수트를 다이어그램으로 구성하는 것이 가능하다[10][12][13].

수트의 구조적 측면은 수트를 이루는 수트와 부수트(sub-SUIT) 관계를 나타낸다. 동적 측면은 수트가 실행되는 시기, 즉 관련된 수트의 수행을 야기시키는 사건에 대한 정보를 나타낸다. 기능적 측면은 수트가 수행되기 위한 데이타를 나타낸다. 구성된 다이어그램은 수트를 나타내는 노드와 속성을 가진 링크로 이루어진 그래프가 된다.

1) 수트의 구조적 측면(structural view)

그림 2의 (a)에서 보인바와 같이, 다이어그램에서 수트는 원으로 표시되고 주수트와 부수트의 관계는 주수트를 나타내는 원과 부수트를 나타내는 원을 연결하는 실선에 의하여 표시된다.



(a) 구조적 측면(structural view)
(b) 동적 측면(dynamic view)
(c) 기능적 측면(functional view)

그림 2. 수트 다이어그램
Fig. 2 SUIT Diagram.

수트는 여러개의 부수트에 의하여 이루어질 수 있지만 상위의 수트를 처리하기 위하여 부수트들이 모두 수행되어야 하는 경우도 있고 부수트 중에서 일부만 수행되어야 하는 경우도 있다. 다이어그램에서 수트의 모든 부수트가 수행되어야 하는 경우를 수트로부터 모든 부수트로 향하는 링크를 가로지르는 호를 그리도록 하여 이를 표현한다. 다이어그램에서 일부의 부수트만 수행되는 경우는 호를 그리지 않는다.

2) 수트의 동적 측면(dynamic view)

수트의 동적인 측면은 수트를 활성화시킬 수 있는 사건(event)을 기술하는 것이다. 사건이 발생했을 때는 어떤 조치가 취해져야 하는데, 여기서는 수트와 태스크의 상태가 시스템의 모든 상황을 표시하기 때문에 수트와 태스크의 현재 상태 변화가 사건이 된다. 이러한 동적인 측면은 2가지의 경우가 있다. 수트는 자신의 수트를 수행하기 위하여 부수트의 실행을 야기(activate)시킬 수 있다. 이 경우에 수트가 필요에 따라 부수트를 야기시키기 때문에 특별한 표현을 필요로 하지 않는다. 또 다른 경우는 임의의 다른 수트 또는 태스크의 상태 변화가 수트를 활성화시키는 사건이 될 수 있다. 이러한 경우를 표현하기 위해서 그림 2의 (b)와 같이 사건을 일으키는 수트 또는 태스크로부터

이 사건에 의하여 수행되는 수트로 화살표를 가진 실선으로 표시한다.

수트의 상태 변화 과정을 그림 3에 보였다. 추론중 사용자 인터페이스를 필요로 하는 일이 수행되는 경우, 이에 관련된 수트가 '활성(active)' 상태가 된다. 만일 이 수트가 부수트를 가지고 있다면 부수트가 모두 수행되는 동안 이 수트는 '대기(wait)'상태에 머물게 된다. 그리고 부수트의 수행 결과와 자신의 수행 결과에 따라 '실현 활성(realized active)'상태 또는 '실현 불가(notrealized)'상태가 된다. '실현 불가'상태가 되었을 경우 부수트들의 상태는 '수면'상태로 된다. '실현 활성' 상태에 있던 수트는 자신과 의존성 관계에 있는 수트가 '활성'상태로 되면 '실현 비활성'상태로 바뀌게 된다. 의존성 관계에 있는 수트가 필요한 기능을 모두 수행하고 '수면(dead)'상태로 바뀌면 '실현 비활성(realized inactive)'상태에 있는 수트는 '실현 활성' 상태로 된다. 역으로 '실현 비활성'상태에 있던 수트가 '수면' 상태로 바뀌면 이 수트와 의존성 관계에 있고 '실현 활성'상태, '실현 비활성'상태에 있던 수트는 '수면'상태로 된다. '실현 비활성'상태 또는 '실현 활성'상태에 있던 수트가 수행해야 할 일을 모두 수행한 경우, 다시 '수면'상태로 돌아간다. 각 상태에 따라 시스템 사용자에게 보여지는 수트의 모습이 바뀐다. 그러므로, 사용자에게 보여지는 내용은 시스템의 수행 상황을 나타낸다.

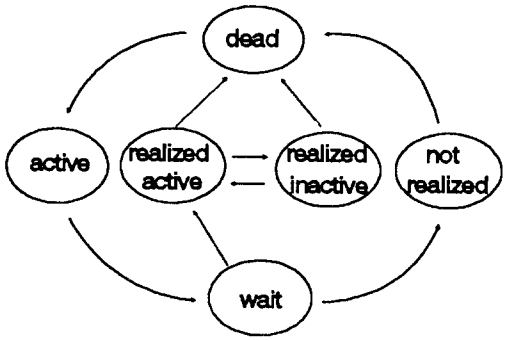


그림 3. 수트의 상태 변이 흐름
Fig. 3. State Transition Flow of SUIT.

3) 수트의 기능적 측면(functional view)

수트의 기능적인 측면은 그림 2의 (c)에서와 같이 수트가 수행에 필요한 자료들에 의해 표시된다. 전통적인 자료 흐름도에서는 프로세스에 필요한 모든 자료를 명세화한다. 이러한 명세화는 변수들을 지역화하여 프로그램의 모듈화를 유도한다. 그러나 전문가 시스템 개발에 있어서는 전문가 시스템 개발 도구에 구성된 지식 표현들이 대개 전역 변수로써 구성되기 때문에 변수 지역화에 대한 의미가 매우 축소된다. 따라서 수트의 기능적 측면은 주로 제어 자료 흐름을 표시하도록 하였으며 수트의 관계에 따라 나타내지 않고 수트에 개별적으로 표현하였다.

4. 수트에 의한 전문가 시스템 사용자 인터페이스

전문가 시스템에서의 사용자 인터페이스는 평면적으로 나타내어진 if-then 규칙에서 필요한 기능을 실행시킴으로써 이루어졌다. 그러므로, 모든 사용자 인터페이스의 제어가 단순히 문제 영역 지식을 표현한 if-then 규칙에 의해서 조절되어야 했다. 즉, 규칙으로 표현된 문제 영역 지식의 추론 결과에 따라 사용자 인터페이스 흐름이 좌우되므로 사용자 인터페이스의 흐름을 파악하기 힘들었고, 제어가 복잡해질 경우 전문가 시스템 개발자도 전체 시스템을 체계적으로 파악하기란 거의 불가능하게 된다. 결국 유지/보수를 위한 비용이 상당히 증가하게 된다.

이 논문에서 제안하는 개발법에서는 문제 분야 지식을 지식 레벨의 태스크에 의해서 표현하고, 수트는 태스크의 상태를 사용자에게 보여주기 위하여 이에 필요한 자료를 덩어리화하여 계층적으로 형성된다. 수트는 여러 개의 부수트를 가질 수 있고 사용자 인터페이스를 위한 일을 추상화하여 나타낼 수 있다. 시스템 구성의 3가지 측면을 수트 다이어그램으로 나타내기 때문에, 전체 사용자 인터페이스 전개 과정을 쉽게 파악할 수 있다. 각 태스크와 수트의 상태에 따라 사용자 인터페이스가 진행된다.

Ⅲ. SUIT 관리기 (SUITMAN)

수트 관리기(SUIT Manager)는 수트에 기술된 내용 및 정의된 일들을 처리한다. 수트 관리기는 수트에 기술된 사용자 인터페이스 내용을 읽어들여 사용자 인터페이스의 가시화를 위해 필요한 함수들을 결정하고, 이를 실행시킨다. 본 논문에서는 X-윈도우상에서 정의된 Motif 라이브러리로부터 제공되는 기능을 호출하여 정의된 수트로부터 자동으로 사용자 인터페이스를 가시화한다.

1. 수트 관리기의 구조

수트관리기의 구조는 그림 4에서 보듯이 크게 사건처리(Event Processing Manager), 표현 처리(Presentation Processing Manager), 수트 상태 결정(State Decision Manager)으로 나뉜다. 사건처리 부분은 사용자로부터의 입력을 받아 사건에 관련

된 속성 값들을 결정하는 일을 한다.

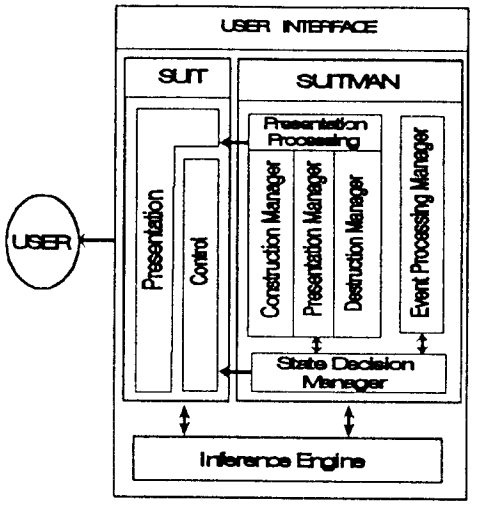


그림 4. 수트 관리기의 구조
Fig. 4. Configuration of SUIT Manager.

사전이 문제분야의 처리를 요구하는 경우인지 또는 단순히 X-라이브러리에 의하여 처리 가능한 경우인지를 판단한다.

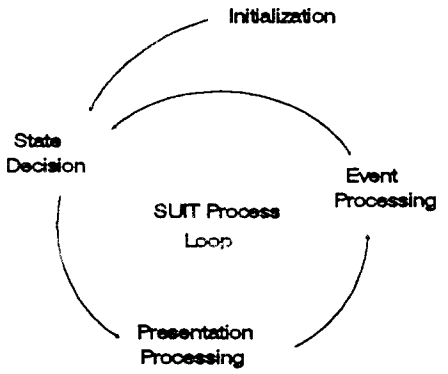


그림 5. 수트망의 수트 관리 흐름
Fig. 5. SUIT Management Flow of SUITMAN.

표현 처리부분은 구성자(Construction Manager), 파괴자(Destruction Manager), 표현 관리자 부분(Presentation Manager)으로 나뉘어진다. 구성자는 수트를 사용자가 직접 볼 수 있는 객체로 만들어주고, 파괴자 부분은 더 이상 사용자로부터의 입력을 받을 수 없는 경우나 사용자에게 정보를 보여 줄 필요가 없을 경우, 가시화된 수트를 화면으로부터 제거한다. 표현 관리자는 현재 수트의 모든 속성값들이 가지는 의미를 함축한 수트의 상태에 따라 사용자에게 보여지는 형태나 모양을 결정한다.

수트 상태 결정 부분은 외부로부터의 사건, 즉 사용자로부터의 입력, 외부 신호, 문제 분야 관련 태스크의 상태 변화 및 문제 분야 관련 데이타의 변화에 관련된 수트를 결정하고 수트에 기술된 상태 변화에 관련된 정보를 참고하여 수트의 상태를 결정한다.

2. 수트관리기의 수트관리 흐름

수트 관리기는 그림 4에서 보듯이 처음 시작할 때의 초기화 단계에서 수트를 초기화한 후, 상태 결정, 표현 처리, 그리고 이벤트 처리의 세 단계를 반복하면서 수트를 관리한다. 초기화 단계에서 수트관리기는 기술된 수트의 속성중 'unknown'인 것을 찾아 초기값으로 설정한다. 'Direction', 'P_Obj_name', 'C_Obj_name'이 'unknown'인 경우, NULL 문자로 설정하고, 'state'나 'newstate'가 'unknown'인 경우, '수면'으로, 'realized_flag'는 'off'로 초기화한다. 초기화는 수트 관리기가 실행될 때 한번만 수행된다.

상태 결정 단계에서 현재 상태의 수트들의 상태를 조사하고 환경이 변화한 경우, 이의 상태를 결정한다. 차례로 '대기', '활성', '실현 활성', '실현 비활성', '실현 불가', '수면'상태를 점검해 본다. 환경의 변화에 영향을 받는 수트만이 점검 대상이 된다.

표현 처리 단계는 구성자, 표현 관리자, 그리고 파괴자로 구성되어 상태의 변화에 따라 사용자에게 보여지는 수트의 모습을 변화시킨다.

이벤트 처리 단계에서는 사용자로부터의 입력을 처리하는 단계이다. 수행중 시스템 내부에서 발생하는 다른 이벤트는 상태 결정 단계에서 수트에 기술된 내용에 따라 처리된다. 사용자로부터의 입력은 이벤트 큐에 저장된다. 수트의 처리가 필요할 경우 이벤트 처리 단계를 빠져 나오게 된다.

위 세 단계를 계속 반복하면서 문제 풀이 과정에서 도출되는 정보를 사용자에게 알린다. 사용자 인터페이스 설계자는 수트 관리기에서 이루어지는 작업에 대하여 상세히 알지 않아도 수트를 만들어 줌으로써 사용자 인터페이스 모듈을 용이하게 개발할 수 있다.

Ⅳ. GUI 개발에의 적용

1. 간단한 예

이 장에서는 수트와 태스크를 이용하여 사용자 인터페이스가 이루어지는 과정을 간단한 예를 통하여 살펴보도록 한다. 간단한 예로서 개인 정보를 입력받는 일은 사용자로부터 개인 정보(주소)를 입력받는 부분과

입력된 주소가 타당한 것인지를 판단하는 부분으로 나뉜다. 이를 태스크를 이용하여 나타내면 그림 6-(a)와 같다. 고객 정보를 입력받는 일을 추상화하여 Determine_Customer 태스크로 나타낸다. 이는 두개의 부태스크 Ask_Customer와 Validate_Data를 갖는다. Ask_Customer는 고객으로부터 입력을 받는 일을 수행하는 태스크이고, Validate_Data 태스크는 '입력된 정보가 타당한가'('실제로 존재하는 주소인가')를 검증한다. Validate_Data 태스크는 Ask_Customer 태스크가 성공적으로 수행되었을 때 수행되는 태스크이다. 3개의 태스크중 Ask_Customer 태스크가 사용자 인터페이스를 필요로 하는 태스크이다.

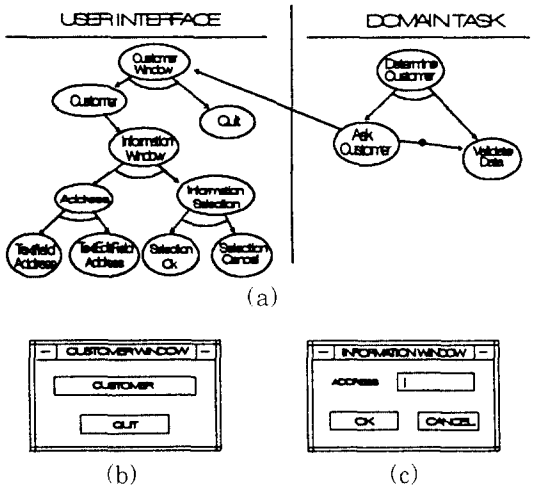


그림 6. 간단한 예의 수트 다이어그램
Fig. 6. A SUIT Diagram and Windows for a Simple Example.

사용자 인터페이스는 수트에 의해 기술된다. 사용자 인터페이스 개발자는 태스크 객체 다이어그램을 사용하여 시스템에서 사용자 인터페이스가 필요한 부분을 파악하고 사용자로부터 입력을 받기 위해 사용자에게 보여지는 윈도우의 모양을 설계한다. 여기서는 사용자로부터 주소를 입력받는 것이므로 그림 6-(b)와 같이 버튼 2개를 갖는 윈도우를 정의한다. 이 윈도우에서 종료하려면 Quit 버튼을 누르고 입력을 하려면 Customer 버튼을 누른다. Customer 버튼을 누르면 주소를 입력할 수 있는 하나의 Text Edit Field와 Ok 버튼과 Cancel 버튼을 갖는 그림 6-(c)와 같은 윈도우가 열린다. Ask_Customer 태스크에서 Customer_Window 수트로 향한 실선 화살표는 Ask_Customer 태스크가 실행되기 위해서 사용자 인터페이스가 요구됨을 나타낸다. 그리고 Customer_Widnow 수트는 Customer 수트와 Quit 수트를 부수트로 가진다. 이는 Customer_Widnow 수트가 사용자에게 보여지기 위해 두 개의 부수트가 실행되어야 함을 나타낸다. 가장 상위 수준의 Customer_Window가 갖는 제어 객체가 문제분야의 Ask_Customer 태스크와의 의존성에 관련된 정보를 갖는다. 이와 마찬가지로 Customer 수트가 갖는 제어 객체가 Customer 수트에 연결된 객체에 관한 정보를 갖는다. 여기서는 Customer 버튼이 눌려졌을 때, 새로운 윈도우가 화면에 보여진다. Customer Window 수트와 Customer 버튼을 연결하는 아크와 Customer Widnow 수트와 Quit 버튼을 연결하는 아크를 가로지르는 호는 Customer_Window 수트가 사용자에게 보여지는 객체로 정의되기 위해서 부수트 모두가 실행되어야 함을 의미한다.

이렇게 시스템의 설계는 문제영역의 태스크와 사용자인터페이스의 수트로 나뉘어 이루어지는데, 먼저 문제영역의 태스크에서 사용자 인터페이스가 필요한 부분을 결정하고 수트 다이어그램을 설계하여 사용자 인터페이스의 제어 흐름과 각 모듈의 계층적 구조를 지정함으로써 각 수트를 제작한다. 이 과정에서 수트 다이어그램은 사용자 인터페이스의 구성과 전개 과정을 그림으로 나타냄으로써 시스템 개발자가 시스템을 전체적으로 파악할 수 있게 하고 시스템 개발자들간의 또는 개발 의뢰자와의 대화 도구로써 사용된다. 시스템 개발자가 수트 다이어그램을 그리고 각 수트의 속성값을 결정하면 수트 관리기에 의해 자동으로 사용자 인터페이스를 실현하게 된다.

2. 회선구성 전문가시스템 프로토타입에서 사용자 인터페이스 구성예

회선 구성 전문가 시스템[14]의 사용자 인터페이스 모듈 일부를 수트와 수트 관리자를 이용하여 구현하였다. 회선 구성 과정을 살펴보면, 사용자의 회선 요구 신청을 받고 이 신청서에 따라 적절한 회선을 구성하고 이를 사용자에게 알리고 망현황의 자료를 갱신하고, 청약자에 관한 자료를 관리하는 작업이 포함된다.

1) 주윈도우의 수트 다이어그램 설계

그림 7에서와 같이, 주윈도우(Main Window)는 전문가 시스템이 동작중에 항상 화면에 출력되어 있는 윈도우로서 CPES 수트에 의해 추상화되었다. 이 윈도우는 크게 사용자로부터 명령을 받는 부분, 망을 화면에 출력하고 이 망에 관련된 내용을 사용자에게 보여주는 부분으로 나뉘어진다.

2) 주윈도우의 수트 설계

앞의 다이어그램에 표시된 CPES 수트를 예로 들어 보면, 그림 8과 같다. CPES 수트는 SUIT 클래스로부터 속성을 상속받고, 제어 객체 CPES_C는 클래스

CONTROL로부터 표현 객체 CPES_P는 클래스 PRESENTATION과 WINDOW로부터 속성을 상속받는다.

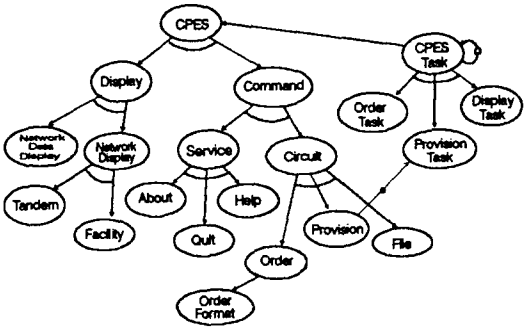


그림 7. 주윈도우의 수트 다이어그램
Fig. 7. The SUIT Diagram of the Main Window.

CPES_P는 윈도우를 나타내는 표현 객체이므로 필요한 속성을 WINDOW로부터 속성을 상속받아야 한다. 다이어그램에서 기술된 내용을 각 객체가 가지는 속성에 필요한 값을 지정하여 줌으로써 수트가 정의된다.

그림 8. 'CPES' 수트의 정의
Fig. 8. A SUIT Object: 'CPES'.

다이어그램을 보면, CPES 수트는 의존 객체(Linked Object name)로서 CPES_Task를 가진다. 이를 나타내는 수트의 속성을 보면, 객체의 방향은 'in'이고 객체의 형태는 'TASK'이다. CPES_Task의 활성화가 CPES 수트를 구동하기 위한 조건임을 나타낸다.

그림 9는 최종적인 회선구성 전문가시스템의 사용자 인터페이스를 보이고 있다. 그림에서의 윈도우는 주윈도우와 사용자로부터 사용자로부터 청약사항을 입력받는 중인 팝업 윈도우를 보인 것이다.

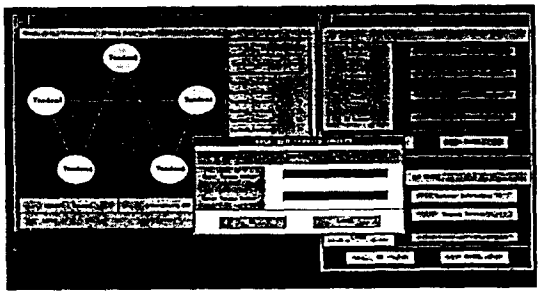


그림 9. 회선구성 전문가 시스템의 사용자 인터페이스
Fig. 9. The User Interface of the Circuit Provisioning Expert System.

V. 결 론

이 논문의 목적은 전문가 시스템 개발시 장애가 되고 있는 사용자 인터페이스 개발을 용이하게 하여 보다 빠른 시작품 개발을 가능하게 하고, 전체 시스템의 사용자 인터페이스 제어 흐름을 기술하는 방법을 제공함으로써 전체 시스템 구조 파악을 쉽게 하여 전문가 시스템의 유지 보수 비용을 줄이고 개발자간의 의사소통을 원활히 하는 것이다. 이를 위하여 사용자 인터페이스를 기술할 수 있는 단위로서 수트를 정의하였고, 전체 시스템의 사용자 인터페이스를 기술하는 방법으로 수트 다이어그램을 제안하였다. 또한 수트 관리기를 개발하여 사용자 인터페이스 개발 시간을 단축하고 소트프웨어 개발 단계 중 구현 단계의 일부를 자동화하였다.

이를 통하여 전문가 시스템 개발자는 문제 분야 영역 설계와 사용자 인터페이스 설계를 독립적으로 분리하여 설계할 수 있었다. 전문가 시스템의 사용자 인터페이스 개발은 수트 다이어그램을 사용하여 사용자 인터페이스 제어 흐름을 결정하고 수트 다이어그램의 각 수트가 갖는 속성에 특정값을 설정하여 주면, 수트 관리기가 기술된 수트의 내용을 읽어들여 사용자 인터페이스를 실현한다. 이것은 설계자는 물론이고 사용자 뿐만아니라 개발자들 또한 쉽게 이해할 수 있으므로 능률적인 정보 교환 수단으로 사용할 수 있다. 향후 연구과제로 지능적 인터페이스를 위한 사용자 모델링 기법과 시스템 문맥 관리에 대한 연구가 이루어져야 할 것이다.

참 고 문 헌

[1] H. R. Hartson and D. Hix, "Human-Computer Interface Development: Concept and Systems for its Management," ACM Computing Surveys, vol.21, no. 1, pp.5-92, March 1989.

[2] Michael E. Atwood, "A Report on the Vail Workshop on Human Factors in Computer Systems," IEEE, Trans. on CG&A, vol.4, no.12, pp.48-66, December 1984.

[3] R. S. Pressman, Software Engineering, McGraw-Hill, Inc., New York, 1992.

[4] Ed Lee, "User Interface Development Tools," IEEE Software, vol.7, no.3, pp. 31-36, May 1990.

[5] D. J. Kasik, "A User Interface Management Systems," Computer Graphics, vol. 18, no.2, pp.99-106, March 1982.

[6] Deborah Hix, "Generations of User-Interface Management Systems," IEEE software, vol.7, no.5, pp.77-87, September 1990.

[7] S. G. Kim, S. H. Kim, C. S. Park and J. Kim, "SUIT-A Schematic User Interface Task for the Specification and the Implementation of Graphic User Interface in an Expert System," Proc. of International Conference on InfoScience '93, pp.678-682, 1993.

[8] M. Green, "The University of Alberta User Interface Management Systems," Computer Graphics, vol. 19, no. 3, pp.205-213, 1985.

[9] Len Bass and Jo lle Coutaz, Developing Software for the User Interface, Addison-Wesley Publishing Company, New York, 1991.

[10] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, New Jersey, 1991.

[11] Joseph W. Sullivan and Tyler, Sherman W., Intelligent User Interfaces, ACM Press, New York, pp. 259-279, 1991.

[12] C. S. Park and J. Kim, "An Expert System Developing Methodology by Using TASK Concepts," Proceedings of the 1st Korea-Japan Joint Conference on Expert System, pp.367-379, 1993.

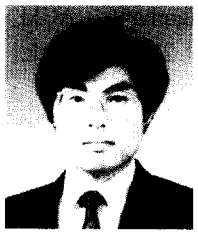
[13] M. Maentylae, "A Modelling System for Top-down Design of Assembled Products," *IBM Journal of Research and Development*, vol. 34, no. 5, pp.636-659, Se- pt. 1990

[14] 김재희, 한국통신 장기기초 연구보고서: 데이타 품질관리 및 자동회선구성 전문가시스템의 구현 방안에 관한 연구, 한국통신, 1993.

저 자 소 개

金 相 吉(準會員)

1968年 8月 12日生. 1992年 2月 한국과학기술원 과학기술대학 전기 및 전자공학과 졸업(공학사). 1994年 2月 연세대학교 대학원 전자공학과 졸업(공학석사) 주관심분야는 전문가 시스템, 사용자 인터페이스, 컴퓨터 그래픽스 등임.



金 成 勳(正會員)

1966年 3月 1日生. 1988年 2月 서강대학교 전자공학과 졸업(공학사). 1990年 2月 연세대학교 대학원 전자공학과 졸업(공학석사). 1990年 3月 ~ 현재 연세대학교 대학원 전자공학과 박사과정 재학중. 주관심분야는 인공지능, 필기인식, 서명검증, 전문가시스템, 사용자 인터페이스 등임.

朴 忠 植(正會員) 第 30卷 B編 第 1號 參照
현재 영동공과대학 전자계산학과 조교수

金 在 熹(正會員) 第 31卷 B編 第 3號 參照
현재 연세대학교 전자공학과 교수