사용성 중심설계를 지원하기 위한 사용자 인터페이스 프로토타입의 생성기법

(A User Interface Prototype Generation Technique Supporting Usage-Centered Design)

김 정 옥 † 유 철 중 †† 장 옥 배 ††
(Jeong-Ok Kim) (Cheol-Jung Yoo) (Ok-Bae Chang)

요 약 웹 환경의 급성장에 따른 애플리케이션의 사용성 중심설계가 부각되고 있다. 본 논문에서는 사용성 중심설계를 기반으로 사용자 인터페이스의 프로토타입을 자동으로 생성하기 위한 개발 단계와 알고리즘을 제안한다. 즉, 요구분석으로부터 사용자 인터페이스의 전이 객체를 모델링하고, 사용자 인터페이스를 생성하기 위한 비즈니스 이벤트의 추상화 모델링 규칙과 알고리즘을 만들었다. 이것은 사용자 인터페이스에서 비즈니스 이벤트들의 가시적 응집도를 높일 수 있고, 미숙한 설계자도 양질의 사용자 인터페이스의 프로토타입을 개발할 수 있도록 지원한다. 또한, 사용자가 인터페이스에 대한 비즈니스 업무의 이해도를 향상시키고, 프로토타이핑의 반복 횟수를 줄일 수 있다.

키워드 : 사용자 인터페이스, 프로토타입, 사용성 중심설계, 모델링, 비즈니스 이벤트

Abstract According to rapid growth of a web environment, usage-centered design is being highlighted. This paper suggests the development step and algorithm to generate a user interface prototype automatically on the basis of usage-centered design. For this purpose, The transition object of the user interface is modeled from requirement analysis, and an abstract modeling rule as well as algorithms of business events are made for the creation of the user interface. Through these processes, visual cohesions of business events become strong and unskilled designers can develop the qualified user interface prototypes. A user's understanding of business tasks can also be improved and prototype iterations reduced.

Key words : User Interface, Prototype, Usage-Centered Design, Modeling, Business Events

1. 서 론

인터넷의 급성장은 소프트웨어의 환경을 웹으로 급속하게 전환시키고, 웹 애플리케이션을 새로운 패러다임으로 등장하게 하였다. 이러한 변화와 함께 사용자 인터페이스도 새로운 룰과 고품질의 입출력이 이루어지는 소프트웨어로 발전하고 있다. Jackob Nielsen은 웹에서 영업의 잘못으로 수십억의 손실이 발생할 수 있는데 그 원인은 사용성 문제라고 추정될 수 있다고 평가하였다[1]. 웹의 특성상 고객이 찾고 있는 것을 쉽게 찾을 수 없으면 고객은 바로 사이트를 이동한다. 이러한 패러다임의 변화와 함께 소프트웨어 엔지니어링의 개념도 사용자 중심설계(user-centered design)에서 사용성 중심설계(usage-centered design)로 발전하고 있다. 따라서 이와 관련된 여러 분야의 연구가 활발하게 진행되고 있다. 애플리케이션의 개발에 있어서 애플리케이션 코드의 48%가 사용자 인터페이스에 그리고 구현 시간의 50%가 사용자 인터페이스의 구현에 소요되고 있다[2,3]. 이와 함께 사용자 인터페이스의 자동생성에 관한 연구가 많이 진행되고 있다. 복잡한 인간과 컴퓨터의 상호작용을 지원하기 위한 사용자 인터페이스는 고객의 요구사항을 수집하고 협상하는 초안물로서 매우 포괄적이고 다방면의 지식을 요구한다[4]. 좋은 사용자 인터페이스의 설계를 위해서는 그래픽 전문가, 요구사항 분석가, 시스템 설계자, 프로그래머, 기술 전문가, 사회 행동과학자, 그리고 업무분야에 따라서 그 분야의 전문가를 필요로 한다[5]. 이렇게 다방면의 전문가를 참여시키는 것은

† 비 회 원 : 대천대학 컴퓨터전기전자계열 교수
kjo3852@dcc.ac.kr
†† 종신회원 : 전북대학교 컴퓨터과학과 교수
cjyoo@chonbuk.ac.kr
okjang@chonbuk.ac.kr
논문접수 : 2003년 5월 12일
심사완료 : 2003년 9월 22일

어렵다. 따라서 양질의 사용자 인터페이스를 자동으로 생성하고 지원할 수 있는 연구가 필요하다. 그러나 대부분의 연구는 인터페이스를 생성할 수 있는 방법에 관한 연구이며, 사용자 인터페이스의 비즈니스 이벤트를 어떻게 구성할 것인가에 관한 연구는 찾기가 어렵다. 따라서 본 논문에서는 고품질의 사용자 인터페이스의 생성을 지원하기 위하여 사용성 중심설계를 기반으로 사용자 인터페이스의 프로토타입을 자동 생성하고, 비즈니스 이벤트의 가시적 응집도가 향상된 사용자 인터페이스를 구성할 수 있는 프로세스와 알고리즘을 제안한다.

본 논문의 구성은 2장에서는 사용자 중심설계와 사용성 중심설계의 차이점을 알아보고, 사용자 인터페이스 생성에 관련된 연구를 살펴본다. 3장에서는 사용성 중심설계의 개발 프로세스를 기반으로 사용성 중심설계에 의한 사용자 인터페이스 프로토타입(User Interface Prototype : 이하 UIP라 함)의 생성 단계를 제안한다. 4장에서는 본 논문에서 제안한 UIP의 생성 단계에 의한 요구분석과 비즈니스 이벤트를 모델링하는 기법과 알고리즘을 논한다. 5장에서는 UIP의 추상화 모델링 단계별 적용 결과와 프로토타입의 생성 결과를 논한다. 6장은 연구결과에 대한 의의와 향후과제를 논한다.

2. 관련연구

사용자 중심설계와 사용성 중심설계의 차이를 논하고, 사용자 인터페이스의 자동 생성 관련연구와 지원도구를 알아본다.

2.1 사용자 중심설계와 사용성 중심설계의 비교

웹 애플리케이션에서 사용성과 사용자의 경험은 시스템의 성공과 실패의 중요한 결정적 요소가 되고 있다. 사용자 인터페이스는 다음의 세 가지 기술의 구현에 있어서 효율적인 지원이 이루어져야 한다. 첫째는 사용자의 요구사항이 정확하게 정의되어야 한다. 둘째는 사용자 인터페이스 설계의 반복에 있어서의 피드백을 위한 빠른 문서 프로토타이핑(Rapid Paper Prototyping)이 이루어져야 한다. 셋째는 프로토타입이나 시스템 작업에 있어서의 문제를 정의하기 위하여 사용성 테스팅이 이루어져야 한다. 그러나 사용자 중심설계는 사용자를 이해하고 그들을 참여시키는 인간의 형이상학적인 측면의 기술요소를 통합하는 것이 부족하다. 즉, 사용자를 이해하고 그것을 설계에 관련시켜서 사용자가 진정으로 원하는 것과 필요로 하는 것을 구분하기가 어렵다. 또한 적절한 설계를 통해서 회피하려고 하는 문제점을 발견하는 데에 있어서도 효율적으로 이루어지기가 어렵다. 이러한 문제점을 보완하기 위하여 사용성 중심설계는 사용자의 성공적인 태스크를 위한 사용성에 초점을 맞추어서 탐구적 모델링이 이루어지고 사용성 검사와 체계적인 프로세스에 의한 공학적인 설계가 이루어진다[6, 7]. 사용성 중심설계에 관련된 연구는 Constantine의 연구가 대부분으로써 이 분야에 관한 더 많은 연구가 요구되고 있다. 따라서 사용성 중심설계의 특성을 이해하기 위하여 사용자 중심설계와 비교하여 운용, 개발, 설계 방법으로 요약하면 표 1과 같다[1].

2.2 사용자 인터페이스 생성 관련연구

사용자 인터페이스의 자동생성에 관련된 연구와 사용자 인터페이스 설계를 지원하는 시뮬레이션에 관련된 연구를 살펴본다.

2.2.1 사용자 인터페이스의 자동생성 관련연구

사용자 인터페이스의 자동생성 연구로는 GENIUS[8], JANUS[9], TRIDENT[10], GUIPS[11]가 있는데, GENIUS는 애플리케이션 도메인을 엔티티 관계모델에서 캡처하고, 뷰와 목록 명세로부터 사용자 인터페이스를 생성한다. JANUS는 객체 모델로부터 사용자 인터페이스의 윈도우를 추출하고, 비 추상화 클래스가 윈도우에 연결된다. TRIDENT는 태스크 분석과 기능적 요구분석에 의한 사용자 인터페이스를 생성하는 방법으로 활동 연결 그래프(Activity Chaining Graph)를 이용하여 데이타와 기능을 상호작용 태스크에 연결하도록 그려진다. GUIPS는 UML을 기반으로 유스케이스 시나리오로부터

표 1 사용자 중심설계와 사용성 중심설계의 비교

구분	사용자 중심설계(User-centered design)	사용성 중심설계(Usage-centered design)
운용방법	· 사용자에 초점 : 사용자의 경험과 만족	· 사용성에 초점 : 태스크 성공을 지원하기위한 개선된 툴
	· 사용자 입력에 의한 운용	· 모델과 모델링에 의한 운용
개발방법	· 실제적 사용자를 포함 - 사용자의 연구 - 설계의 참여 - 사용자 피드백 - 사용자 테스팅	· 선택적 사용자 포함 - 탐구적 모델링 - 모델링의 확인 - 사용성 검사
설계방법	· 반복적 프로토타이핑에 의한 설계	· 모델링에 의한 설계
	· 매우 가변적, 비정형, 명기되지 않는 프로세스	· 체계적, 충분하게 명세된 프로세스
	· 시운전과 에러에 의한 설계, 진화	· 공학적 설계

표 2 사용자 인터페이스 자동생성 연구의 비교

구분	TRIDENT	JANUS	GUIPS
도메인모델	- 데이타와 태스크 분석모델	- 객체 모델	- UML기반의 객체분석 모델
설계방법	- 태스크와 기능 요구분석에 의한 사용자 인터페이스 추출	- 객체모델로부터 사용자 인터페이스를 추출	- 시나리오로부터 요구분석에 의한 사용자 인터페이스 추출
명세방법	- 데이타와 기능요구 명세	- 데이타 구조적 명세	- 전이 객체의 데이타 명세
특징	- 상호작용 태스크에서 애플리케이션 분석과 사용자 인터페이스에 의한 태스크 속성 결정 - 활동 연결 그래프(데이타와 기능의 상호작용) - 사용자 인터페이스의 정적인 면을 다룸	- 비추상화 클래스가 사용자 인터페이스에 전달 - 사용자 인터페이스에 관련이 없는 속성과 메소드는 프로세스 과정에서 무시 - 사용자 인터페이스의 동적인 면은 다루지 않음 - 태스크 분석 무시	- 시나리오로부터 사용자와의 상호작용 인터페이스의 모델링 - 객체 전이 그래프(인터페이스와의 상호작용) - 사용자 인터페이스의 프로토타입 생성 프로세스

폼으로 전환되고, 사용자 인터페이스와 애플리케이션 도메인의 형식을 명세하였다. 사용자 인터페이스의 자동생성에 관한 최근연구의 특성을 살펴보면 표 2와 같이 요약된다.

2.2.2 사용자 인터페이스의 설계에 관련된 시뮬레이션 연구

사용자 인터페이스의 설계를 위한 시뮬레이션 연구는 STATEMATE[12], SCR[13], ASMOOS[14] 등의 다양한 방법과 도구에 의해서 지원되고 있다. 이러한 연구의 특징을 살펴보면 STATEMATE는 구조적, 기능적, 행위적 관점에서 시스템을 묘사하기 위한 그래픽과 도식어를 제공하고, 자동화 코드 및 검증 시뮬레이션을 지원한다. SCR은 요구정의를 위하여 테이블에 의한 표기법을 제안하고, 시뮬레이션과 자동 에러 방지 도구를 제안한다. ASMOOS는 시나리오 세트로부터 상태기계 다이어그램을 구성하기 위한 알고리즘을 나타내고 있다. 이러한 연구의 특성을 살펴보면 표 3과 같이 요약된다.

3. 사용성 중심설계에 의한 UIP의 생성 프로세스

Constantine & Lockwood가 제안한 사용성 중심설계의 개발 프로세스를 알아보고, 이 프로세스를 기반으로 사용성 중심설계에 의한 UIP의 생성 단계를 제안한다.

3.1 사용성 중심설계의 개발 프로세스

사용성 중심설계의 논리적 프로세스는 사용자인 인간 액터와 시스템 액터로 분류하고 역할 모델, 태스크 모델, 콘텐츠 모델이 서로 조화를 이루면서 개발되고 연결되어진다. 실제적 근원이 되는 사용자의 역할인 액터는 사용자의 역할을 지원하는 태스크 케이스로 연결되고, 각 페이지와 폼 그리고 다른 상호작용 비즈니스 이벤트는 내부적으로 관련된 태스크 케이스의 묶음을 지원하는 추상화 프로토타입에 대응된다. 설계자는 태스크 케이스를 지원하고 있는 범위에서 특정 단계를 구현하는 추상화 컴포넌트로부터 행위 버튼, 링크, 테이블, 디스플레이와 다른 생성물을 추출한다. 결국 이러한 태스크 케이스는 사용자가 실행할 수 있는 역할을 지원한다. 사용성 중심설계의 추상적 모델은 사용자가 원하는 작업을 충분히 달성할 수 있도록 지원하여 가장 단순하게 시스템을 체계적으로 설계하기 위하여 사용된다. 사용성 중

표 3 사용자 인터페이스 설계지원 연구의 비교

연구구분	특 징
STATEMATE	- 시스템 묘사를 위한 그래픽과 도식어 제공 - 기능적, 구조적, 행위적 뷰 도식 - 자동화 코드 생성 및 검증 시뮬레이션 지원 - 사용자 인터페이스의 생성은 지원되지 않음
SCR	- 요구정의를 테이블에 의한 표기법 제안 - 시뮬레이션과 자동에러 방지를 위한 도구제공 - 형식명세 모델은 고전적 상태머신 모델 - 시뮬레이션 툴은 사용자 인터페이스 통합을 위해서 제공 - GUI 빌더를 사용한 매뉴얼 구성
ASMOOS	- 시나리오 셋으로부터 상태머신 상태 다이어그램을 구성하는 알고리즘 - 애니메이션 설계에 의한 설계 - 시나리오 생성이 사용자 인터페이스를 통해서 지원되고 매뉴얼화 - 통합 상태머신을 통합하는 동안 새로운 시나리오가 생성됨

심설계에 관련된 개발 프로세스는 Constantine & Lockwood의 연구가 대부분인데, 이 분야에 관한 더 많은 연구가 필요하다. 다음은 Constantine & Lockwood[1]가 제안한 사용성 중심설계의 프로세스로서 개발 단계별 역할은 다음과 같다.

1) 예비단계

① 본질적인 목적과 예상 : 비즈니스와 사용자 목적을 명료화한다. 특징, 편리성, 내용물, 가능성을 예상하고 추측한다.

② 탐구 모델링 : 질문, 모호성, 위험하고 불확실한 영역을 정의한다.

2) 반복 프로세스 개발 단계

③ 역할 모델링 : 모든 사용자의 역할 목록을 작성하고 우선순위를 부여한다. 초기목표 서브 셋을 선정한다.

④ 태스크 모델링 : 모든 태스크의 목록을 작성하고 우선순위를 부여한다. 초기 목표 서브셋을 선정한다.

⑤ 태스크 클러스터링 : 유사성에 의한 모든 태스크를 그룹핑한다. 총체적인 네비게이션의 구조를 입안한다.

⑥ 기본 설계 : 상호작용과 가시적 골격을 입안한다. 심미적 설계를 검토하고 교정한다.

⑦ 추상화 프로토타이핑 : 서브 셋으로 선정된 태스크를 지원하는 상호작용 관계를 위한 콘텐츠 모델을 개발한다.

⑧ 상호작용 설계 : 선정된 상호작용 관계를 위하여 상세화한 사용자 인터페이스 설계를 개발한다.

⑨ 가시화 프로그래밍 구축 : 사용자 인터페이스로 설계된 부분의 프로그램을 구축한다.

3.2 사용성 중심설계에 의한 UIP의 생성 단계

Constantine & Lockwood[1]가 제안한 사용성 중심설계의 프로세스를 기반으로 사용성 중심 설계에 의한 UIP의 생성 단계를 제안한다.

사용성 중심설계는 웹 환경의 변화에 따라서 사용자의 요구를 빠르게 대응하고, 사려 깊은 프로토타핑을 빠르게 반복할 수 있도록 하기 위한 개발 단계이다[15, 16]. 본 논문에서는 이러한 패러다임을 지원하기 위한 일환으로 사용성 중심설계의 개발 프로세스를 연구하였다. 본 논문에서 제안한 UIP의 개발 단계는 그림 1과 같이 요구분석 역할 모델링, 태스크 모델과 클러스터링, 컨텐츠의 비즈니스 이벤트를 위한 추상화 프로토타입 모델, 그리고 사용자 인터페이스를 생성하는 단계로 구성된다[19,20]. 제안한 UIP의 사용성 중심설계에 의한 프로토타이핑의 추상적 운용모델은 역할 모델, 태스크 모델, 추상화 프로토타핑 모델이 있다. 역할 모델은 사용자가 시스템에 관해서 해야 할 역할의 특성을 추출하

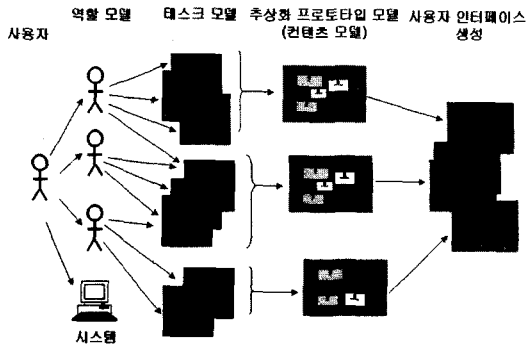


그림 1 사용성 중심설계에 의한 UIP 생성 단계

며 사용자가 시스템에서 가질 수 있는 관계를 표현한다. 태스크 모델은 시스템으로 달성해야 할 작업 구조를 나타내며, 태스크 케이스를 결합하여 태스크 케이스의 관련성을 매핑한다[17,18]. 추상화 프로토타이핑 모델은 사용자 인터페이스의 내용과 구조를 나타내고, 사용자 인터페이스의 비즈니스 이벤트를 추상화하여 사용자와의 상호작용이 어떻게 구성되는지를 나타내는 추상화 프로토타입이다. 제안된 UIP의 개발 단계와 생성물을 요약하면 다음과 같다.

① 역할 모델링 : 모든 사용자의 역할 목록을 작성하고 우선순위를 부여한다(유스케이스 다이어그램, 순차다이어그램).

② 태스크 모델링 : 모든 태스크의 목록을 작성하고 태스크의 순서를 부여한다(상태 다이어그램, 태스크 전이 목록).

③ 태스크 클러스터링 : 관련성에 의한 모든 태스크를 그룹핑한다. 총체적인 네비게이션의 구조를 입안한다(객체 전이 그래프, 상호작용 객체 전이 그래프).

④ 추상화 프로토타입 모델링 : 선정된 태스크를 지원하는 상호작용 관계를 위한 컨텐츠(추상화 프로토타입) 모델을 설계한다(필드 추상화 모델, 태스크 추상화 모델, 트랜잭션 추상화 모델, 폼 추상화 모델).

⑤ 가시화 프로그래밍 구축 : 사용자 인터페이스로 설계된 부분의 프로그램을 구축한다(비즈니스 이벤트, 컨텐츠, 폼의 생성).

제안된 UIP 프로세스의 개발단계는 Constantine & Lockwood[1]가 제안한 프로세스에서 태스크 클러스터링 단계와 기본설계 단계를 통합하였으며, 추상화 프로토타이핑 단계와 상호작용 설계 단계를 통합하였다. 이것은 상호작용에 의한 유사성과 가시적 응집도를 기반으로 모델링이 자동으로 이루어지도록 함으로써 클러스터링과 기본설계가 통합하여 이루어지도록 하였다. 또한, 상호작용을 기반으로 역할, 태스크, 트랜잭션 단위의

추상화 모델링이 이루어지고, 필드 추상화 모델링에서 구체적인 비즈니스 이벤트가 설계되어 사용자 인터페이스의 프로토타입을 생성함으로써 추상화 프로토타이핑 단계와 상호작용 설계 단계를 통합하였다. 이러한 개발 단계의 통합에 의한 자동화 모델링은 개발이 용이하고 개발기간을 단축시킨다.

4. 사용성 중심설계에 의한 UIP의 생성기법과 알고리즘

3.2절에서 제안된 UIP를 생성하기 위하여 제안된 프로세스의 단계별로 모델링 규칙을 만들고, 생성 알고리즘을 적용 예와 함께 논한다.

4.1절의 요구분석에 의한 역할 모델링 단계는 사용자와의 인터뷰에 의한 유스케이스 다이어그램(이하 Use-CaseD이라 함)을 작성하고, 태스크 모델링이 용이하도록 하기 위하여 사용자 인터페이스와의 상호작용 순서를 쉽게 모델링할 수 있는 순차 다이어그램(이하 SeqD이라 함)을 작성한다. 4.2절의 태스크 모델링 단계는 SeqD를 이용하여 상태 다이어그램(이하 StateD이라 함)이 작성되는데 StateD는 모든 전이 객체를 알 수 있도록 작성되고, 태스크 전이 목록이 작성된다. 다음은 태스크를 통합하기 위하여 태스크 클러스터링이 이루어지는데, 클러스터링 단계에서는 객체 전이 그래프가 작성되고, 사용자와의 상호작용이 이루어지지 않는 전이 객체는 그래프에서 삭제한다. 4.3절의 추상화 프로토타이핑 단계는 사용자 인터페이스에서 비즈니스 이벤트들의 가시적 응집도를 향상시키기 위한 단계로서 비즈니스 이벤트를 추상화하여 모델링이 이루어진다.

4.1 역할 모델링

4.1.1 유스케이스 다이어그램의 작성

사용자와의 인터뷰에 의한 요구분석 단계로서 모든 사용자의 역할 목록을 작성하고 유스케이스를 작성한다. 그림 2와 같이 UseCaseD를 작성하고, UseCaseD는 유스케이스를 실체화하기 위하여 객체들 간의 연관성을

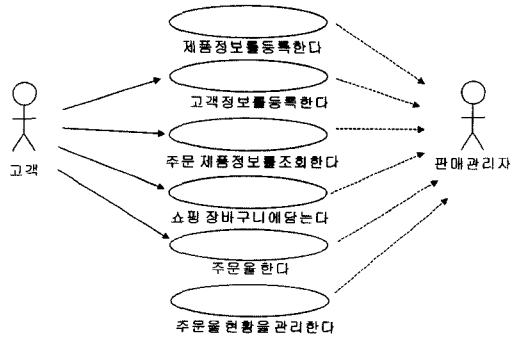


그림 2 요구분석 역할 모델링을 위한 UsecCaseD

중심으로 시나리오에서 객체를 도출하고, 객체들 간의 유기적인 관계와 주고받는 메시지를 모델링한다. 이것은 사용자 인터페이스에서 비즈니스 이벤트의 기능적 응집도를 높이기 위하여 사용자의 역할을 모델링하는 단계이다.

4.1.2 순차 다이어그램의 작성

사용자 인터페이스의 객체를 추출하여 객체들 간의 연결과 메시지를 파악하고, 진행순서를 모델링하기 위하여 UseCaseD로부터 SeqD를 작성한다. 이것은 협력 다이어그램으로 나타낼 수도 있지만 객체의 전이 순서를 나타내기에는 SeqD가 용이하므로 SeqD를 사용하여 작성하였다[20]. 이것은 사용자 인터페이스에서 비즈니스 이벤트의 순차적 응집도를 높여주기 위한 방법이며, 매뉴얼의 워크 플로우를 쉽게 이해할 수 있도록 지원한다. 그림 3은 객체간의 연결과 메시지를 나타내는 SeqD를 보여주고 있다.

4.2 태스크 모델링

태스크 모델링은 SeqD를 이용한 상태 다이어그램을 작성하고, 태스크 전이 목록을 작성한다. 다음은 태스크를 통합하기 위하여 객체 전이 그래프가 작성되고, 사용자와의 상호작용이 이루어지는 전이 객체를 추출한 객체 전이 그래프가 작성된다.

4.2.1 태스크 상태도 모델링

네비게이션 구조를 입안하기 위하여 태스크의 모델을 검토한다. 태스크의 상태 전이를 일으키는 이벤트와 상태 전이의 결과인 액션으로 인스턴스화되어 생성에서 소멸 상태까지 인터페이스의 전이 객체를 모델링하는 단계이다. 객체의 전이 과정을 StateD로 변환하고, 객체의 생명주기를 상호작용 관계의 StateD로 작성하여 전이 순서를 부여한다. 그림 4는 전이 객체의 생명주기와 전이 순서를 StateD로 나타낸 것이다. 이것은 비즈니스 이벤트의 통신적 응집도를 높이기 위하여 전이 태스크의 단위로 모델링한다.

그림 4의 StateD는 전 단계에서 작성된 SeqD를 이용하여 생성한다. 이것은 UML을 이용하여 SeqD를 CollD로 전환할 수 있다. 따라서 전 단계에서 작성된 SeqD를 CollD로 전환하고, 관련연구[21]의 CollD-To-StateD 전환 알고리즘을 이용하여 CollD를 StateD로 전환한다. 이때 상세한 태스크의 모델링이 이루어지고 태스크의 상세한 전이 객체의 목록도 함께 작성된다. 전이 목록은 사용자 인터페이스의 비즈니스 이벤트를 생성하기 위하여 모든 전이 목록을 작성해야한다. 표 4는 이때 작성된 전이 태스크의 상세한 전이 목록 명세를 나타낸다.

4.2.2 태스크 클러스터링

StateD에서 태스크 단위의 객체 전이 그래프를 만든

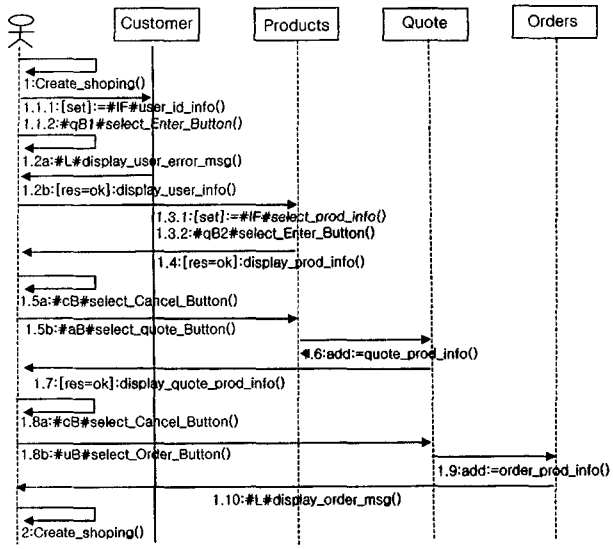


그림 3 요구분석에서 사용자 인터페이스의 객체를 추출하기 위한 SeqD

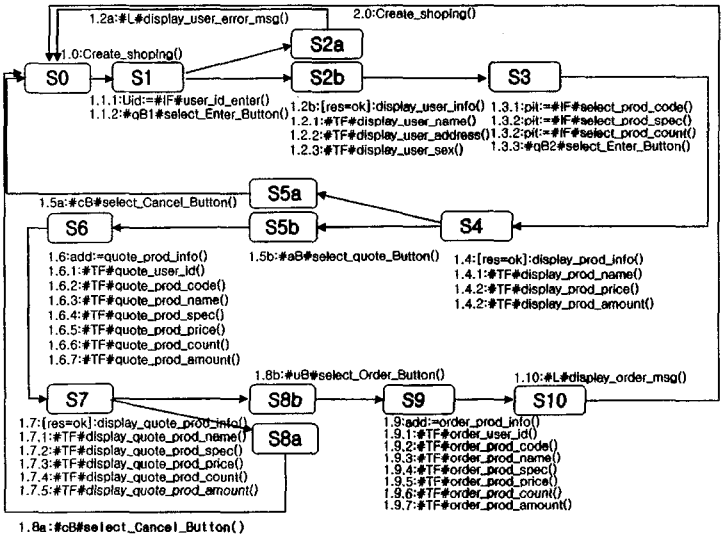


그림 4 태스크 클러스터링을 위한 SeqD-To-StateD

표 4 태스크의 전이 목록

태스크명	상세 전이 목록
user-info	고객ID, 성명, 주소, 주민번호, 나이, 직업, 고객암호
product-info	제품ID, 제품명, 가격, 단위, 용도, 규격
quote-info	고객ID, 견적수량, 단가, 세금, 견적금액, 견적일자
quote-item	품목ID, 제품명, 규격, 수량, 할인율
order-info	주문ID, 수량, 단가, 세금, 금액, 주문금액, 주문일자
order-item	품목ID, 제품명, 규격, 수량, 할인율

다. 객체 전이 그래프의 객체는 사용자 인터페이스, 시스템 인터페이스, 장치 인터페이스로 이루어진다. 이러한 인터페이스들 중에서 사용자와 상호작용하는 사용자 인터페이스의 전이 객체만을 추출하기 위하여 비 상호작용 전이 객체를 삭제한다.

1) 객체 전이 그래프 생성

그림 4에서 만들어진 StateD로부터 M. Elkoutbi 등 [11]이 제안한 객체 전이 그래프(Graph Transition: GT)를 생성하는 방법과 알고리즘을 이용하여 객체 전이

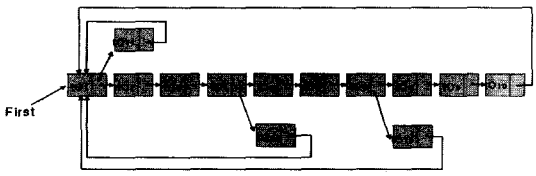


그림 5 StateD로부터 생성된 객체 전이 그래프

```
// StateD 에서 전이 객체 노드리스트의 생성:
ObjectNodeList = sd.TransitionObjects()
// 전이객체의 링크리스트 생성:
LinkList = 0;
For each tOi ∈ ObjectNodelist
  S = sd.originalState(tOi)
  // 입력받은 전이 객체로부터 인스턴스 리스트 받음
  ObjectList = sd.InputTransitionObjects(s)
  // 상태 s 에서 들어온 전이로부터 객체리스트 받음
  For tOi ∈ ObjectList
    LinkList.addLink(tOi, tOi)
  // StateD 의 최초의 상태 s 인경우에
  If S.type == firstState
    Ss=S.superState()
    // 최초의 상태 s 의 전이 객체리스트 받음
    ObjectList = sd.inputTransitionObjects(Ss)
    For tOi ∈ ObjectList
      LinkList.addLink(tOi, tOi)
    End If
  // 결합이나 분기의 전이 상태가 일어난 경우
  If (S.type==andState) or (S.type==orState)
    ObjectList = S.TransitionObjectsInside()
    // 결합이나 분기의 전이 상태가 일어난 경우에 전이 객체리스트를 받음
    For tOi ∈ ObjectList
      LinkList.addLink(tOi, tOi)
  End If
End For
//초기 노드 리스트의 생성:
firstNodeList = 0;
OLFS = sd.firstStates()
For each s ∈ OLFS
  OTO = s.outputTransitionObjects()
  //초기의 노드 리스트의 연결
  firstNodeList = firstNodeList ∪ OTO
End For
```

그림 6 객체 전이 그래프의 생성 알고리즘

그래프를 만들어내는 단계이다. StateD의 전이는 그림 5에서 GT의 전이 노드를 나타내고, 링크는 전이를 실행할 때 순서를 나타낸다. 전이 tO_1(Transition Object: tO)이 전이 tO_2로 전달되는 상태는 전이는 tO_1과 tO_2을 나타내는 노드들 사이의 관계를 링크로 나타내어진다.

그림 6은 M. Elkoutbi등이 제안한 알고리즘으로서 이해하기 쉽도록 일부 변수의 표현만을 수정한 것이다. 이 알고리즘은 객체지향 개념을 이용한 "DOT" 표기법을 사용하여 작성하였다[21-23]. 이 알고리즘은 주어진 StateD의 *sd*로부터 GT의 *ObjectNodeList, LinkList, FirstNodeList*를 얻는 방법을 상세화한 것이다. 이것은 유스케이스를 SeqD로 변경하고, SeqD를 StateD로 전환하여, 최종적으로 전이 알고리즘을 이용한 객체 전이 그래프를 생성하는데 그림 5는 이 결과를 나타낸다. GT의 중간노드를 *ObjectNodeList*, 전이의 연결노드를 *LinkList*, 시작노드를 *FirstNodeList*로 나타낸다. GT의 *ObjectNodeList*는 StateD의 전이 리스트에 대응되기 때문에 쉽게 얻어진다. GT의 *LinkList*는 전이객체 tO로부터 시작된 상태인 모든 전이를 각각의 전이로 연결되도록 정의함으로써 얻어진다.

2) 상호작용 전이 객체의 추출

사용자 인터페이스에 직접적으로 영향을 주지 않는 모든 전이 객체를 제거하는 단계이다. 즉, 사용자 인터페이스의 상호작용 전이 객체만을 추출한다.

이것은 사용자 인터페이스의 비즈니스 이벤트가 되는 전이 객체만을 추출하여 사용자 인터페이스를 모델링하기 위한 단계이다. M. Elkoutbi등이 제안한 관련연구[11]의 알고리즘을 사용하여 상호작용의 객체 전이 그래프를 만들어낸다. 그림 5의 오퍼레이션 결과는 그림 7을 생성한다.

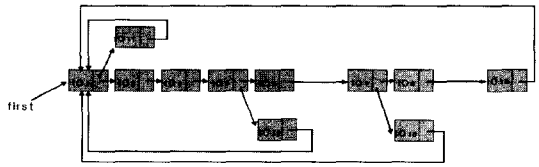


그림 7 비 상호작용의 전이 객체가 삭제된 객체 전이 그래프

4.3 추상화 프로토타입 모델링

본 절에서는 사용자 인터페이스에서 비즈니스 이벤트의 가지적 응집도를 높이기 위하여 필드 추상화, 태스크 추상화, 트랜잭션 추상화, 폼 추상화 등의 4단계로 추상화하고, 모델링 규칙과 알고리즘을 설명한다.

4.2절에서 추출한 사용자 인터페이스의 전이 객체를 이용하여 사용자 인터페이스의 비즈니스 이벤트들을 구성하기 위한 모델링 단계이다. 비즈니스 이벤트의 모델링을 위하여 전이 객체의 유사성, 관련성, 전이 단위에 따라서 사용자 인터페이스의 세부 객체들을 분석하였다. 연구에서 객체들의 유형을 전이 단위로 분류하여 추상화하고, 클러스터링함으로써 이벤트 객체가 순차적, 기능적, 통신적 응집도를 높여 주게 한다[24]. 비즈니스 이벤트의 응집도 향상에 관한 상세한 내용은 각각의 추상화 모델링 단계에서 설명한다.

이 연구 결과로 제안된 비즈니스 이벤트의 추상화 모델링은 그림 8과 같이 4단계로 이루어지고 다음과 같이 요약된다. 1단계의 필드 추상화 모델링은 전이 객체를 필드 추상화 알고리즘에 의해서 비즈니스 이벤트의 유형을 모델링한다. 2단계의 태스크 추상화는 태스크의 전

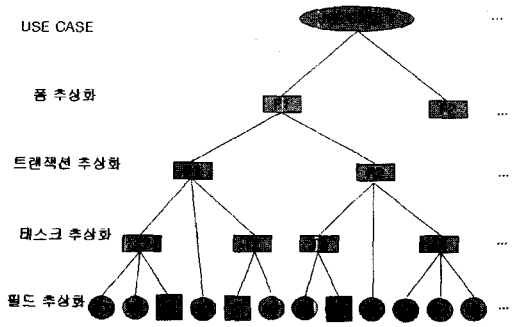


그림 8 비즈니스 이벤트들의 추상화 모델링 계층구조

이 단위로 객체들을 클러스터링하는 모델링 단계이다. 3단계는 트랜잭션 단위로 클러스터링하는데 입력전이-컨트롤-출력전이의 단위로 구성된다. 4단계의 폼 추상화 단계는 폼을 분할하는 단계로서 유스케이스를 몇 개의 폼으로 분할할 것인가를 모델링한다.

4.3.1 필드의 추상화 모델링

필드의 추상화 모델링은 User Interface Field Label Block(UIFLB)을 나타내는 노드가 있는 직선 그래프로 구성되어 이루어진다. 그림 9의 tO_2는 그림 7의 객체 전이 그래프를 상세화한 것을 나타내며, 이렇게 추상화된 단위를 UIFLB라고 정의한다. 이것은 전이 노드의 순서를 구성하는 GT의 부 그래프이고, 자동 입력 필드 등의 비즈니스 이벤트를 설계한다. 즉, 텍스트, 라디오 버튼, 콤보박스, 체크 버튼 등의 비즈니스 이벤트를 설계할 수 있는 필드의 추상화를 위한 오퍼레이션이다. UIFLB의 필드 추상화를 위한 규칙은 다음과 같다.

규칙 1 : 하나의 필드에 제한된 인스턴스의 갯수를 갖는 필드는 UIFLB이다.

규칙 2 : 하나의 필드에 입력할 수 있는 인스턴스의 개수가 일정하지 않으면 UIFLB가 아니다.

규칙 3 : 최대 7개 이하의 인스턴스를 가질 수 있는 항목은 라디오버튼을 사용할 수 있는 필드 추상화 레이블이다(G. Miller의 '7±2 chunks' 이론에 근거[25]).

규칙 4 : 8개 이상의 인스턴스를 가질 수 있는 필드는 콤보박스를 사용할 수 있는 필드의 추상화 레이블이다.

규칙 5 : 선택의 인스턴스를 입력할 수 있는 항목은 체크버튼을 사용할 수 있는 필드 추상화 레이블이다.

그림 10의 알고리즘은 이러한 규칙을 적용하여 만들어진 UIFLB의 추상화 알고리즘을 나타낸다. 그림 7의 사용자 인터페이스의 전이 객체를 순차적으로 읽어서 전이 객체의 *S.type*과 이미 정의된 *UIFLB.type*의 규

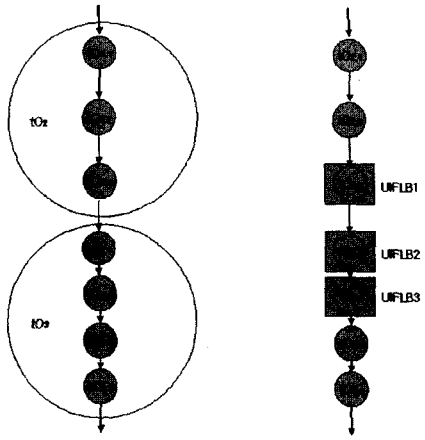


그림 9 필드의 추상화 알고리즘을 적용한 GT

```
Let ObjectNodelist ={tO1,tO2,,..., tOn} be the set of Objects of transition from GT
Let UIFLB = {radio, check, combo,... } be the Objects of User Interface Item Label Block
k = 1
For each tO ∈ ObjectNodelist
  S = StateAttributeFromObjectNodelist(tOi)
  If S.type == UIFLB.type then
    BlockNodeList.add(tOk)
    //블록 전이 객체 노드의 생성(블록의 유형, 인스턴스의 수, 객체의 길이 등록)
    //블록 객체노드(tOk)의 링크의 추가
    LinkList.addLink(tOi+1, tOk)
    LinkList.addLink(tOk, tOi)
    LinkList.deleteLink(tOi+1, tOi)
    //블록 전이 객체노드(tOk)의 링크의 추가
    ObjectNodeList.updateBlock(tOi)
    //전이 객체에 UIFLB 블록타입의 정의
    k = k + 1
  End If
  i = i + 1
End for
```

그림 10 필드의 추상화 모델링 알고리즘

칙을 비교하고, UIFLB의 비즈니스 이벤트의 유형을 결정하여 *BlockNodeList*를 추가하고, *LinkList*도 추가한다. *ObjectNodeList.updateBlock(tO, tOi)*에서는 *BlockNodeList*에 UIFLB라는 비즈니스 이벤트 타입을 설계하여 설계 정보를 등록한다.

4.3.2 태스크의 추상화 모델링

태스크의 추상화 모델링은 User Interface Task Label Block(UITLB)을 나타내는 노드가 있는 GT로 구성된다. 하나 이상의 전이 객체 필드를 갖는 태스크의 전이 단위는 UITLB로 클러스터링 된다. 즉, 이벤트 발생시 전이되는 입력 또는 출력 항목이 2개 이상 존재할 때 Block Label을 만들어서 전이되는 태스크 단위로 클러스터링 한다. 그림 11은 태스크 모델링의 예를 보여준다. 이것은 사용자 인터페이스에서 전이 단위의 비즈

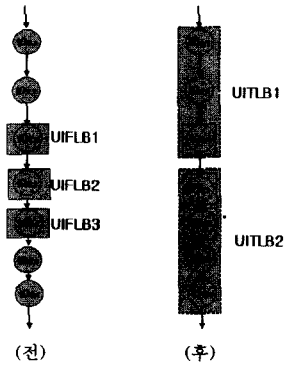


그림 11 태스크의 추상화 모델링 알고리즘을 적용한 GT

```
Let ObjectNodelist ={tO1, tO2,..., tOn} be the set of Objects of transition from GT
Let UITLB= {uitlb1, uitlb2, ..., uitlbn} be the Objects of User Interface Task Label Block
// UITLB 블록번호의 초기값 부여
k = 1
For each tO ∈ ObjectNodelist
  S = TaskAttributeFromObjectNodelist(tOi)
  //전이 객체(tOi) 속성을 S에 받음
  If QueueEmpty()
    SaveFirstObjectLinkOfBlock()
    //UITLB 블록의 첫 번째 전이 객체 노드 링크를 저장
  else if S.TaskAttribute == Queue.TaskAttribute
    j = j +1
    //UITLB 블록내의 객체수의 번호 증가
    Queue[++rear] = tOi
    //같은 UITLB 블록으로 정의될 전이객체를 큐에 저장
  else if j > 1
  // 큐에 저장된 전이 객체가 2개 이상이면 객체 블록을 생성
    BlockNodeList.add(tOk)
    // UITLB 블록객체 노드의 추가(전이 객체의 수, 전이 객체의 최대길이 포함)
    //UITLB 블록객체 노드(tOk)의 링크의 추가

    LinkList.addLink(ITOL, tOk)
    LinkList.addLink(tOk, tOs)
    LinkList.deleteLink(ITOL, tOs)
    For each queue(tOj) ∈ Queue
      ObjectNodeList.update(tOi)
      // UITLB 번호를 정의하여 큐에서 출력
      tOj = queue[++front]
    End For
    // UITLB 블록번호의 증가
    k = k + 1
  else
  // UITLB가 아닌 전이 객체를 블록을 정의하지 않음(전이 객체가 1개 일때)
    tOj = queue[++front]
  End If
  SaveFirstObjectLinkOfBlock()
  //UITLB 블록의 첫 번째 전이객체 노드 링크를 저장
End For
//UITLB 블록 노드 추가를 위한 첫 번째 객체노드 링크를 기억시킴
SaveFirstObjectLinkOfBlock()
{ j = 1
  //UITLB 블록의 첫 번째 전이객체 노드 링크를 저장
  ITOL = LinkList.InputObjectTransationLink(tOi)
  tOs  = ObjectNodelist(tOi)
  Queue[++rear] = tOi
  //읽어온 객체를 큐에 저장
}
```

그림 12 태스크의 추상화 모델링 알고리즘

니스 이벤트들의 기능적 응집도를 높여준다. 태스크 전이 단위의 클러스터링을 위한 규칙은 다음과 같다.

규칙 1 : 하나 이상의 연속되는 전이 태스크는 UITLB를 구분하는 블록이다.

규칙 2 : 하나 이상의 레코드를 갖는 출력 UITLB는 StringGrid 블록이다.

규칙 3 : 하나 이상의 연속되는 입력필드를 갖는 노드는 UITLB이다.

규칙 4 : 하나 이상의 연속되는 출력필드를 갖는 노드는 UITLB이다.

이러한 규칙을 적용하여 생성된 사용자 인터페이스의 전이 객체 태스크 블록에는 블록의 생성에 필요한 필드의 최대 길이와 개수를 등록한다. 이것은 비즈니스 이벤트를 구현할 때 Block Label 생성에 필요한 블록 사이즈의 산출에 사용한다. 그림 12는 이러한 규칙에 따라서 생성되는 UITLB의 알고리즘을 나타낸다. *SaveFirst-ObjectLinkOfBlock()*은 순차적으로 읽어들인 전이 객체를 큐에 저장하고, 저장된 전이 객체의 태스크와 새로 읽어들인 전이 태스크를 비교한다. 태스크가 다르고 저장된 객체가 1개 이상이면 *BlockNodeList.add(tOk)*에서 전이 블록 객체를 추가하고, *LinkList.addLink*에서 링크를 연결한다. 이때 큐에 저장되어 있는 전이 객체는 모두 같은 UITLB이 된다. *ObjectNodeList.update(tOi)*은 어떤 전이 객체 블록에 속하는지를 알 수 있도록 UITLB의 번호와 UITLB 내의 전이 객체의 순서를 부여하는 과정을 나타내고, *queue[++front]*은 큐에 저장된 모든 전이 객체는 큐에서 출력시킨다. 그림 11의 왼쪽은 상세한 전이 객체를 나타내고, 오른쪽은 왼쪽의 객체 전이 그래프를 전이 추상화 알고리즘에 의하여 생성한 UTILB을 나타낸다.

4.3.3 트랜잭션 추상화 모델링

트랜잭션 추상화 모델링은 User Interface tRrans-action Label Block(UIRLB) 나타내는 노드가 있는 직선 그래프로 구성된다. UIRLB는 입력-컨트롤-출력 객체로 구성되는 클러스터링 단계이다. 사용자의 요구(입력)와 응답(출력)을 하나의 단위로 Block화하여 전이 객체들을 클러스터링한다. 이것은 전이 객체들을 트랜잭션 단위로 클러스터링함으로써 사용자 인터페이스의 순차적 응집도를 높여준다. 트랜잭션 단위의 클러스터링을 위하여 다음과 같이 규칙을 정의하였다.

규칙 1 : 반드시 입력필드-버튼-출력필드로 구성되고, 입력필드는 이전 트랜잭션의 출력필드를 사용할 경우 생략될 수 있다.

규칙 2 : 하나 이상의 입력 필드와 하나 이상의 출력필드를 가질 수 있다.

규칙 3 : 하나 이상의 연속되는 입력필드의 첫 번째

노드가 UIRLB의 시작이다.

규칙 4 : 하나 이상의 연속되는 출력필드의 마지막 노드가 UIRLB의 끝이다.

이러한 규칙에 의해서 UIRLB를 생성하고, 이것은 그림 12의 태스크 추상화 알고리즘을 적용할 수 있다. 태스크 추상화 알고리즘에서는 전이 태스크의 속성을 비교하지만 트랜잭션 추상화 알고리즘에서는 UTILB의 마지막 출력 전이 객체가 올 때까지 읽어서 큐에 저장한다. 큐에 저장된 전이 객체는 UIRLB의 번호와 UIRLB 내의 전이 객체의 순서를 부여하여, 큐에 저장된 모든 객체를 큐에서 출력시킨다. 이 트랜잭션 추상화 알고리즘은 그림 12의 태스크 추상화 모델링 알고리즘의 사용이 가능하므로 생략하도록 한다. 그림 13의 왼쪽은 읽어들인 객체 전이 그래프를 나타내고, 오른쪽은 왼쪽의 객체 전이 그래프를 트랜잭션 추상화 알고리즘에 의하여 생성한 UIRLB를 나타낸다.

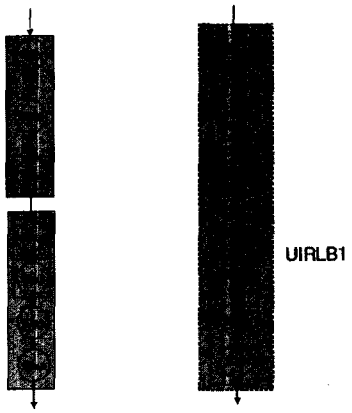


그림 13 트랜잭션 추상화 알고리즘을 적용한 GT

4.3.4 폼 추상화 모델링

폼 추상화 모델링은 User Interface Page Block (UIPB)를 나타내는 노드가 있는 직선 그래프로 구성된다. 사용자 인터페이스에 있는 비즈니스 이벤트들을 폼으로 분할하기 위한 추상화 단계이다. 입출력 필드가 20개를 초과하거나(인간환경 공학의 척도), 하나의 입력에서 출력 폼이 선택(or_state)인 경우, 인터럽트와 같이 사용자에게 명확하게 인식시킬 필요가 있는 경우에 출력 폼을 분할한다. 폼을 분할하기 위한 규칙은 다음과 같다.

규칙 1 : 입출력 객체가 20개 이상이고, 동일 태스크가 아니면 페이지를 분할한다.

규칙 2 : 요구에 대한 응답이 2개중 선택인 경우 페이지 블록으로 구성된다.

규칙 3 : 인터럽트인 경우 새로운 페이지 블록으로 구성한다.

규칙 4 : 20개 항목이 넘어도 동일 태스크 추상화 블록이면 페이지를 분할할 수 없다.

규칙 5 : UIRLB가 폼으로 분할될 때는 출력 UITLB를 다른 폼으로 분할한다.

폼의 분할 알고리즘도 그림 12의 태스크 추상화 알고리즘을 적용할 수 있다. 그림 13에서 생성된 전이 객체를 읽어 들여서 큐에 저장하고, 저장된 전이 객체의 수를 카운트한다. 폼 분할 조건에 맞으면 UIPB의 번호와 UIPB 내의 전이 객체의 순서를 부여하여, 큐에 저장된 모든 객체를 큐에서 출력시킨다. 이렇게 등록된 정보는 폼의 생성시 필요한 정보들이다. 이와 같이 폼 추상화 알고리즘도 그림 12의 태스크 추상화 모델링 알고리즘의 사용이 가능하므로 생략하도록 한다. 그림 14에서 왼쪽은 읽어들인 객체 전이 그래프를 나타내고, 오른쪽은 왼쪽의 객체 전이 그래프를 폼 추상화 알고리즘에 의하여 생성한 UIPB를 나타낸다.

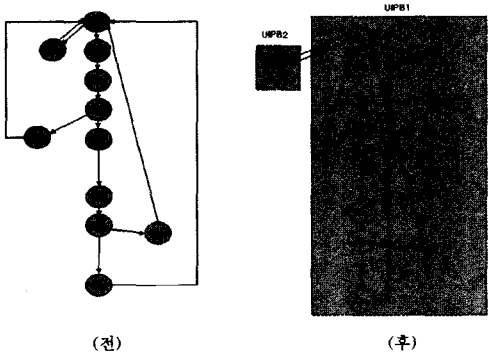


그림 14 폼 추상화 알고리즘을 적용한 GT

5. 사용성 중심설계에 의한 UIP 생성기법의 적용결과

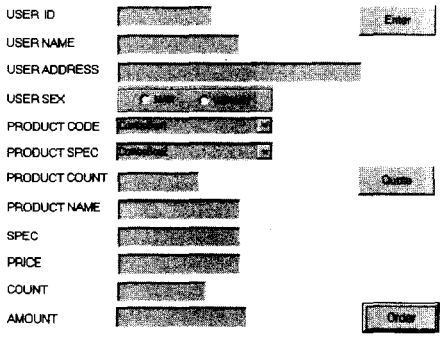
추상화 모델링에 의한 프로토타입의 생성 기법을 단계별로 적용하여 그 결과를 살펴보고, 기존연구와 UIP의 생성 알고리즘에 의한 최종결과를 분석한다.

5.1 UIP의 추상화 모델링 단계별 적용결과

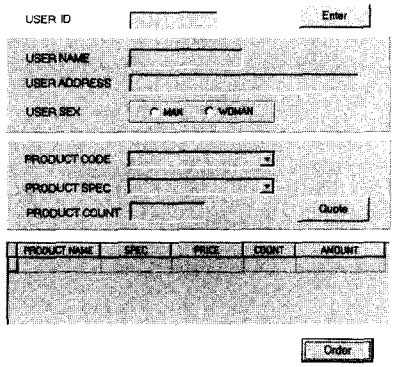
본 논문에서 제안한 추상화 모델링 결과를 UIP의 생성 알고리즘에 적용하여 그 결과를 분석한다. 그림 15는 본 논문에서 제안한 추상화 모델링의 단계별 적용 결과를 나타낸다. 이 단계별 생성 결과의 특성을 살펴보면 다음과 같다.

1) 1단계 : 필드 추상화 모델링 결과

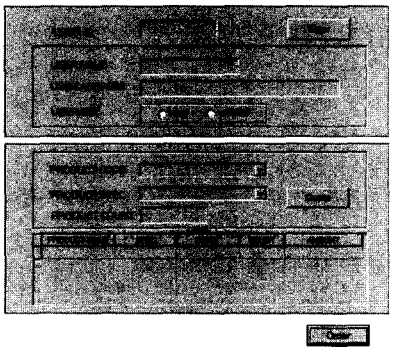
전이 객체 필드의 인스턴스 데이타를 조사하여 사용자가 이해하기 쉽고, 자동 입력을 제공할 수 있도록 하기



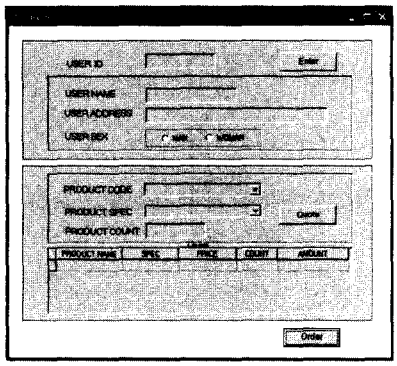
(a) 1단계 : 필드 추상화 모델링 결과



(b) 2단계 : 태스크 추상화 모델링 결과



(c) 3단계 : 트랜잭션 추상화 모델링 결과



(d) 4단계 : 폼 추상화 모델링 결과

그림 15 UIP의 추상화 모델링 적용 결과

위한 단계이다. 즉, 텍스트, 라디오 버튼, 콤보박스 등을 모델링하여 자동으로 제공하기 위한 비즈니스 이벤트의 추상화 모델링한다. 이것은 사용자 인터페이스에서 비즈니스 이벤트의 기능을 효과적으로 모델링하여 비즈니스 이벤트의 기능적 응집도와 재사용성을 높여준다.

2) 2단계 : 태스크 추상화 모델링 결과

전이 단위에 의한 추상화 모델링 단계로써 클래스 객체 흐름의 이해를 돕기 위하여 태스크의 전이 단위로 클러스터링한다. 이것은 비즈니스 이벤트를 태스크 단위로 클러스터링하고, 레이블화하여 사용자 인터페이스에서 비즈니스 이벤트의 통신적 응집도와 기능적 응집도를 높여준다.

3) 3단계 : 트랜잭션 추상화 모델링 결과

이것은 트랜잭션 단위의 추상화 모델링 단계로써 태스크의 입력-컨트롤-출력으로 구성한다. 사용자 인터페이스의 비즈니스 이벤트를 트랙잭션 단위로 클러스터링하여 사용자의 이해를 쉽게 한다. 즉, 태스크의 입력-컨트롤-출력을 단위로 클러스터링하여 레이블화 함으로써 사용자가 인터페이스의 트랜잭션 단위를 쉽게 이해할 수 있도록 한다. 이것은 트랜잭션 단위로 전이 순서를 구성하여 사용자 인터페이스에서 전이 객체들의 순차적 응집도를 높여준다.

4) 4단계 : 폼 추상화 모델링 결과

사용자 인터페이스에서 비즈니스 이벤트들을 이용하여 폼을 분할하기 위한 추상화 모델링이다. 입출력 항목이 20개를 초과하거나 하나의 입력에서 출력 폼이 선택(or_state)인 경우, 그리고 인터럽트와 같이 사용자에게 명확한 인식이 필요한 경우에 출력 폼을 분할한다. 효과적인 폼의 분할은 비즈니스 이벤트의 기능적 응집도를 높여준다[24].

5.2 사용성 중심설계에 의한 UIP의 생성

추상화 프로토타핑에 의하여 모델링된 객체 전이 그래프를 이용하여 사용자 인터페이스를 가시화하기 위한 UIP를 프로그래밍하는 단계이다.

그림 16은 UIP의 생성 알고리즘을 나타낸다. 이 오퍼레이션의 *generateObjectPrototype()*은 사용자 인터페이스의 필드 비즈니스 이벤트를 생성하고, *generate-BlockPrototype()*은 사용자 인터페이스의 Block Label

```
Let IO be the set of interface objects in the system,
Let UC={uc1, uc2,..., ucn} be the set of use cases of the system,
Let UB={ub1,ub2,,..., ubn} be the set of user interface blocks of the system,
Let UO={uo1,uo2,,..., uon} be the set of user interface objects of the system,
For each io in IO
  For each uc in UC
    For each ub in UB
      If io.UserInterfaceBlock(ubi) then
        // 추상화된 사용자 인터 페이스 블록의 생성
        ui = io.getGTforUserInterfaceBlock(ubi)
        ui.generateBlockPrototype()
      else If io.userInterfaceObject(uoi) then
        For each uo in UO
          //사용자 인터페이스 객체 생성
          ui = io.getGTForUserInterfaceObject(uoi)
          ui.generateObjectPrototype()
        End for
      End If
    End for
  End for
  io.generateCompletePrototype()
End for
```

그림 16 UIP의 생성 알고리즘

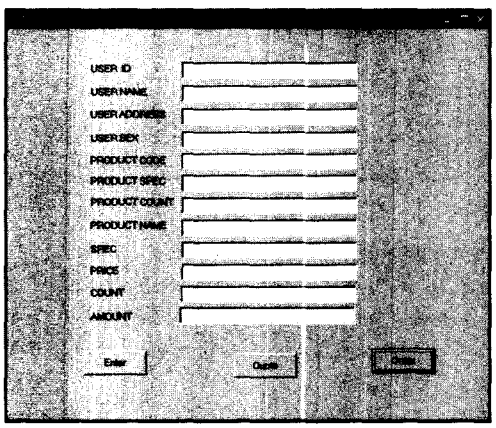


그림 17 GUIPS의 프로토타입 생성결과

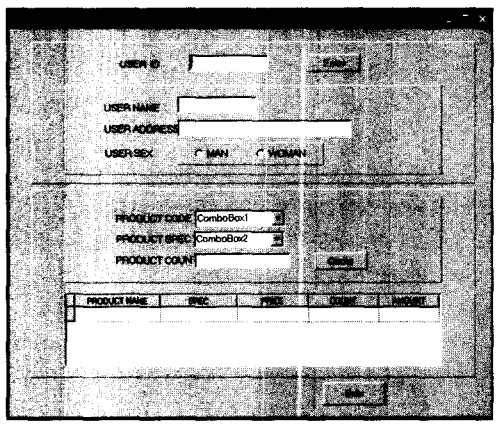


그림 18 제안한 UIP의 프로토타입 생성결과

을 생성한다. *generateComplete Prototype()*은 여러 유스케이스로부터 하나의 애플리케이션을 생성하는 프로토타입을 통합한다.

그림 17은 기존연구 GUIPS[11]의 실행결과를 나타내고, 그림 18은 본 논문에서 제안된 UIP의 생성 알고리즘을 이용하여 생성된 사용자 인터페이스의 실행결과를 나타낸다. 제안한 UIP가 기존 연구와의 차이점은 추상화 모델링에 의한 사용자 인터페이스의 내용과 구조를 어떻게 구성하고, 사용자 인터페이스에서 사용자와의 상호작용을 고려하여 비즈니스 이벤트의 응집도를 향상을 시키기 위한 모델링이 이루어진다는 것이다. 이렇게 설계된 UIP의 전이 객체 클래스는 Rational Rose를 이용하여 클래스 속성을 정의하여 그에 상응하는 골격 코드가 자동으로 생성할 수 있다[26,27]. 사용성 중심설계의 프로세스에 의해서 생성된 사용자 인터페이스 프로토타입은 비즈니스 이벤트들의 응집도가 높은 폼을 만들고, 프로토타핑의 반복을 줄일 수 있도록 지원한다. 프로토타입과 프로그래밍 코드의 생성은 분석자와 프로그래밍 개발자를 지원하여 프로그램의 개발 기간과 인력을 줄일 수 있다.

6. 결론 및 향후 연구과제

사용성 중심의 설계 방법은 사용자를 지원하고 사용성을 향상시키기 위한 개발 방법론이다. 웹 애플리케이션의 사용성 중심설계에 대한 패러다임은 신속성과 사용성에 있다. 따라서 본 연구는 빠른 프로토타입의 생성과 고품질의 사용자 인터페이스의 설계에 목적을 두고 있으며, 연구 결과에 대한 의의를 살펴보면 다음과 같다.

첫째, UML기반에서 비즈니스 이벤트의 추상화 모델링에 의한 사용자 인터페이스 프로토타입의 개발을 자동화한다.

둘째, 웹을 기반으로 사용성 중심설계에 의한 사용자 인터페이스 프로토타입의 생성 프로세스를 개발하였다.

셋째, 객체와 태스크를 기반으로 사용자 인터페이스를 모델링 함으로써 골격코드를 자동으로 생성하여 프로그래밍을 지원한다.

넷째, 사용자 인터페이스 설계의 자동화 모델링을 제공함으로써 미숙한 설계자도 양질의 프로토타입을 생성할 수 있도록 지원한다.

다섯째, 역할 모델링, 태스크 모델링, 클러스터링에 의한 비즈니스 이벤트의 기능적, 순차적, 통신적 응집도가 높아져서 사용자 인터페이스의 기능성과 사용성을 향상시킨다[24].

여섯째, 추상화 모델링으로 가시적 응집도가 향상된 사용자 인터페이스를 제공함으로써 사용자와의 커뮤니케이션 에러를 감소시키고, 통합 개발 프로세스의 프로

토타이핑 반복 횟수를 감소시킨다.

이렇게 자동화된 사용자 인터페이스의 프로토타입에서 객체의 생성, 비즈니스 이벤트의 추상화, 코딩 작업은 수작업보다 개발시간을 단축시킬 수 있다. 그러나 본 논문은 첫째, 환경상의 문제로 프로토타입의 반복횟수를 감소시키는 다수의 검증실험은 하지 못하였다. 둘째, 태스크의 내부전이 객체를 이용한 빈의 자동설계에 관한 연구는 직접적인 인터페이스가 아니므로 포함하고 있지 않다.

사용자와 컴퓨터 사이의 상호작용에서 생성되는 데이타 플로우를 다루는 컴퓨터 프로그램이 사용자 인터페이스 부분이다. 애플리케이션 코드에서 사용자 인터페이스가 차지하는 비중은 매우 높고, 구현하는 것도 매우 어렵다. 또한, 사용자 인터페이스 프로그램은 사용이 쉬워질수록 개발은 더욱 어려워진다. 따라서 사용자 인터페이스 프로그램의 개발을 자동으로 지원할 수 있는 사용자 인터페이스의 생성에 관한 계속적인 연구가 필요하다. 향후 연구과제로는 첫째, 추상화 모델링의 가시적 응집도를 검증할 수 있는 연구가 필요가 있다. 둘째, 사용자 인터페이스의 효과적인 모델링에 의한 가시적 응집도를 향상시키기 위한 더 많은 연구가 필요하다. 셋째, 추상화 모델링을 위한 컨트롤의 유형과 추출 방법에 관한 깊이 있는 연구가 필요하다.

참 고 문 헌

[1] L. L. Constantine and L.A.D. Lockwood, "Usage Centered Engineering for Web Applications," *IEEE Software*, Vol. 19. No 2. pp. 42~50, March-April 2002.

[2] M. D. Lozano, P. Gonzalez, and I. Ramos, "User Interface Specification and Modeling in an Object Oriented Environment for Automatic Software Development," *Technology of Object-Oriented Languages and Systems*, 2000. TOOLS 34. Proceedings. 34th International Conference, pp. 373~381, 2000.

[3] 이상근외, *"소프트웨어 재사용 시스템을 지원하는 사용자 인터페이스 구축기의 설계 및 구현"*, 정보처리학회 논문지, 제2권 제3호, pp. 324~334, 1995.5.

[4] 김성한외, *"XML 기반의 e-비즈니스 문서 생성을 위한 폼 생성 시스템"*, 정보처리학회 논문지 D, 제9권 제4호, pp. 713~722, 2002.8.

[5] Leszek A. Maciazek, *"Requirements Analysis and System Design,"* Addison Wesley, Mass., 2001.

[6] C. Stray, "TADEUS : Seamless Developement of Task-Based and User-Oriented Interfaces," *IEEE Transactions on*, Vol. 30, No 5, pp. 509~525, September 2000.

[7] A. Parush, "Usability Design and Testing," *ACM Interactions*, Vol. 8, No. 5, September-October 2001.

[8] C. Janssen, A. Weisbecker, and U. Ziegler, "Generating User Interfaces from Data Models and Dialogue Net Specifications," *Proc. of the Conference on Human Factors in Computing Systems (CHI'93)*, Amsterdam, The Netherlands, pp. 418~426, April 1993.

[9] H. Balzert, "From OOA to GUIs: The Janus System," *IEEE Software*, Vol. 8, No 9, pp. 43~47, February 1996.

[10] F. Bodart, A.-M. Hennebert, J.-M. Leheureux, I. Provot, and J. Vanderdonckt, "A Model-based Approach to Presentation: A Continuum from Task Analysis to Prototype," *Proceedings of the Eurographics Workshop on Design, Specification, Verification of Interactive Systems*, Carrara, Italy, Focus on Computer Graphics, Springer-Verlag, Berlin, pp.77~94, June 1994.

[11] M. Elkoutbi, I. Khriss, and R.K. Keller, "Generating User Interface Prototypes from Scenarios," 1999. Proceedings. *IEEE Inernational Symposium on*, pp. 150~158, 1999.

[12] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, and M. Trakhtenbrot, "STATEMATE: A Working Environment for the Development of Complex Reactive Systems," *IEEE Transactions on Software Engineering*, Vol. 16, No 4, pp. 403~414, April 1990.

[13] C. Heitmeyer, J. Kirby, B. Labaw, and R. Bharadwaj, "SCR*: A Toolset for Specifying and Analyzing Software Requirements," *Proc. of the 10th Annual Conference on Computer-Aided Verification, (CAV'98)*, Vancouver, Canada, pp. 526~531, 1998.

[14] K. Koskimies, T. Systa, J. Tuomi and T. Mannisto, "Automatic Support for Modeling OO Software," *IEEE Software*, Vol. 15, No 1, pp. 42~50, January-February 1998.

[15] A. Knight and N.. Dai, "Objects and the Web," *IEEE Software*, Vol. 19, No 2, pp. 51~59, March-April 2002.

[16] J. Nielsen, *"Designing Web Usability : The Practice of Simplicity,"* New Riders Publishing, Reading, Mass., 1999.

[17] L. L. Constantine and L.A.D. Lockwood, *"Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design,"* Addison Wesley, Mass., 1999.

[18] M. V. Harmelan, *"Object Modeling an User Interface Design,"* Addison Wesley, Reading, Mass., 2001.

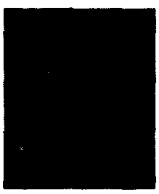
[19] I. Jacobet, G. Booch, and J. Rumbaugh, *"The United Software Developement Process,"* Addison Wesley Longman, Reading, Mass., 1999.

[20] I. Jacobet, G. Booch, and J. Rumbaugh, *"The Unified Modeling Language User Guide,"* Addison

Wesley, Reading, Mass., 1999.
[21] S. Schöberger, R. K. Keller, and I. Khriss, *"Algorithmic Support for Model Transformations in Object-Oriented Software Development,"* Concurrency and Computation: Practice and Experience, Vol. 13, No 5. pp. 351~383, 2001.
[22] E. Horowitz, S. Sahni, and S. Rajasekaran, *"Computer Algorithms C++,"* Computer Science Press, Reading, 1997.
[23] E. Horowitz, S. Sahni, and D. Mehta, *"Fundamentals of Data Structures in C++,"* W. H. Freeman and Company, Reading, 1995.
[24] D. A. Ruble, "Practical Analysis & Design for Client/Server & GUI Systems," Prentice-Hall, Inc., Reading, Mass., 1997.
[25] G. A. Miller, "The Magical Number Seven Plus or Minus Two : Some Limits on Our Capacity to Process Information," *Psychological Reviews*, Vol. 63, pp. 81~87, 1956.
[26] B. Shannon, *"Java 2 Platform Enterprise Edition,"* Addison Wesley, Reading, Mass., 2000.
[27] E. Roman, *"Mastering Enterprise JavaBeans,"* John Wieley & Sons, Reading, Mass., 1999.

김 정 옥

1985년 전북대학교 전산통계학과 졸업(이학사). 2000년 전북대학교 대학원 컴퓨터정보학과 졸업(이학석사). 2004년 전북대학교 대학원 전산통계학과 졸업(이학박사). 1985년~1990년 포항제철 전산시스템부 근무. 1990년~1996년 한신생명보험 정보시스템부. 1996년~1999년 전북도시가스 전산실장 2002년~현재 대천대학 컴퓨터전기전자계열 초빙교수. 2000년 정보처리기술사. 관심분야는 소프트웨어 개발 프로세스, 소프트웨어 메트릭스, GUI, UML, 소프트웨어 품질, 정보검색, 웹 소프트웨어



유 철 중

1982년 전북대학교 전산통계학과 졸업(이학사). 1985년 전남대학교 대학원 계산통계학과 졸업(이학석사). 1994년 전북대학교 대학원 전산통계학과 졸업(이학박사). 1982년~1985년 전북대학교 전자계산소 조교. 1985년~1996년 전주기전여자대학 전자계산과 전임강사~부교수. 1997년~현재 전북대학교 자연과학대학 컴퓨터과학과 전임강사~부교수. 관심분야는 소프트웨어 개발 프로세스, 소프트웨어 품질, 컴포넌트 소프트웨어, 소프트웨어 메트릭스, 소프트웨어 에이전트, GNSS, GIS, 교육공학, 인지과학

장 옥 배

1973년 고려대학교 졸업(학사,석사). 1988년 산타바바라대 대학원(Ph. D.) 1974년~1980년 죠지아 주립대. 오하이오 주립대 박사과정 수료. 1980년~현재 전북대학교 자연과학대학 컴퓨터과학과 교수. 관심분야는 소프트웨어공학, 전산교육, 수치해석, 인공지능