객체 재사용성 향상을 위한 레거시 시스템 인터페이스 기반 객체추출 기법

(An Object Extraction Technique for Object Reusability Improvement based on Legacy System Interface)

이 창 목 † 유 철 중 †† 장 옥 배 ††
(Chang-Mog Lee) (Cheol-Jung Yoo) (Ok-Bae Chang)

요 약 본 연구는 객체 재사용과 재공학을 위해 레거시 시스템의 인터페이스 정보로부터 의미 있는 정보를 추출하고 새로운 시스템에 통합될 수 있도록 하기 위한 기존 레거시 시스템의 인터페이스에 기반한 객체추출 기법을 제안한다. 본 논문에서 제안하는 객체추출 기법은 인터페이스 사용 사례 분석 단계, 인터페이스 객체 분할 단계, 객체구조 모델링 단계, 객체 모델 통합 단계 등 4단계로 구성되어 있다. 인터페이스 사용 사례 분석 단계는 인터페이스 구조 그리고 레거시 시스템과 사용자간의 상호작용 등의 정보를 획득하는 단계이다. 인터페이스 객체분할 단계는 인터페이스 정보를 의미 있는 필드들로 구분하는 단계이며, 객체구조 모델링 단계는 인터페이스 객체들 간의 구조적 관계와 협력 관계를 파악하여 모델링하는 단계이다. 마지막으로 객체 모델 통합 단계는 객체 단위의 단위 모델들을 통합하여 추상화된 정보를 포함한 상위 수준의 통합 모델을 유도하는 단계다. 객체추출 기법에 의해 생성된 객체 통합 모델은 역공학 기술자들의 레거시 시스템 이해와 레거시 시스템의 정보를 새로운 시스템에 적용하는데 있어 좀 더 용이한 효율성을 제공한다.

키워드 : 객체 분석 프로세스, 역공학, 인터페이스 객체, 통합 모델

Abstract This paper suggests a technique, TELOR(Technique of Object Extraction Based on Legacy System Interface for Improvement of Object Reusability) for reuse and reengineering by analyzing the Legacy System interface to distill the meaningful information from them and disassemble them into object units which are to be integrated into the next generation systems.

The TELOR method consists of a 4 steps procedure: 1) the interface use case analysis step, 2) the interface object dividing step, 3) the object structure modeling step, and 4) the object model integration step. In step 1, the interface structure and information about the interaction between the user and the Legacy System are obtained. In step 2, the interface information is divided into semantic fields. In step 3, studies and models the structural and collaborative relationship among interface objects. Finally, in step 4, object model integration step, integrates the models and improves the integrated model at a higher level.

The objects integration model created through TELOR provides a more efficient understanding of the Legacy System and how to apply it to next generation systems.

Key words : Object Analysis Process, Reverse Engineering, Interface Object, Integrated Model

1. 서 론

객체지향 패러다임이 제시된 이래 오늘날 대부분의 시스템을 개발하는데 객체지향 패러다임이 폭 넓게 적용되어가고 있는 추세이다. 그 이유는 객체지향 패러다임이 갖는 재사용성과 새로운 시스템에 자연스럽게 통합되어질 수 있는 유연함 때문이다. 그러나 아직도 대부분의 기업에서는 비 객체지향적 또는 객체지향 언어를 사용하여 시스템을 개발하였으나 객체지향 개념이 정확히 적용되지 않은 레거시 시스템을 그대로 사용하고 있다. 왜냐하면, 기업에서 사용하고 있는 현 시스템이 가장 안정적이기 때문이다. 그러나 이러한 레거시 시스템은 급변하는 업무 환경에 적합하게 대처할 수 없는 경

† 학생회원 : 전북대학교 대학원 컴퓨터통계정보학과
cmlee@chonbuk.ac.kr
†† 종신회원 : 전북대학교 컴퓨터과학과 교수
cjyoo@chonbuk.ac.kr
okjang@chonbuk.ac.kr
논문접수 : 2004년 4월 21일
심사완료 : 2004년 9월 6일

우가 대부분이다[1].

따라서, 이러한 레거시 시스템에서 나타나는 몇 가지 문제점들을 보면, 첫째, 레거시 시스템은 개발 문서가 없는 경우이거나 형식에 그친 문서들뿐이다. 이는 레거시 시스템에 대해서 알려진 지식이 불충분하고, 그 결과 시스템에 대한 이해가 부족하게 되어 그 시스템의 성장과 발전이 어렵게 되는 원인이 된다. 둘째, 유지보수가 어렵다는 것이다. 특히, 기능(function)을 보다 관리가 용이한 구성 요소(component 또는 object)들로 분할할 수 없다. 셋째, 대부분의 기업에서 사용하고 있는 레거시 시스템들은 80년대에 개발된 기술을 사용하고 있는 경우가 대부분이다. 이는 호스트(host)중심의 폐쇄형 시스템이므로 시스템 통합이 어렵다[1].

현재까지 레거시 시스템에 대한 문제점을 해결하려는 많은 전략 중에는 레거시 시스템과는 관계없이 처음부터 현 상황에 알맞은 시스템을 새로이 개발하는 방안도 있다. 그러나 레거시 시스템은 차세대 시스템과 통합되어 새로운 기능을 생성해 낼 수 있는 자산으로서 레거시 시스템을 폐기하는 방법은 적절하지 않다[2,3]. 따라서, 레거시 시스템의 의미 있는 정보(semantics)를 추출해내는 것이 중요하다. 이러한 레거시 시스템으로부터 의미 있는 정보를 추출해내는 방법중 역공학을 이용한 방법이 대표적인 예라 할 수 있다. 역공학 방법의 핵심은 레거시 시스템으로부터 단지 의미 있는 정보만을 추출하는 것이 아니라, 추출된 의미 있는 정보를 보다 상위수준의 추상화로 유도하는 것이다[4].

본 논문에서는 레거시 시스템의 자원 중에서 인터페이스 정보로부터 차세대 시스템에 적용될 수 있는 의미 있는 객체 정보를 추출해서 새로운 시스템과 통합될 수 있도록 하기 위한 역공학 프로세스 즉, TELOR (Techniques to Extract Object Based on Interface of Legacy System for Object Reusability)를 제안한다. 이를 위해서 레거시 시스템의 인터페이스로부터 보다 상위수준의 개념적 추상화를 얻기 위해서 역공학 개념을 이용한다. 또한, 객체지향적 기법은 레거시 및 차세대 애플리케이션 시스템을 동시에 일관성 있게 정의할 수 있도록 하기 때문에 객체구조 모델링 기법이 적용된다. 객체구조 모델링은 레거시 시스템의 인터페이스로부터 객체와 객체들 간의 구조를 파악하기 위한 것으로써 객체구조 모델링을 통해 파악된 의미 있는 객체 정보들은 레거시 시스템을 재사용 하는데 보다 용이할 것이다.

레거시 시스템으로부터 인터페이스에 관한 지식을 습득하고 습득된 지식을 이용하여 객체지향 모델을 생성해내는 것이 본 논문의 목적이다. 인터페이스는 최종 사용자를 위한 사용자 인터페이스이다. 다시 말하면, 애플리케이션을 사용하는데 있어 최종 사용자로 하여금 가장 일반적으로 사용되는 공통된 수단이다[5]. 애플리케이션 시스템 대부분의 지식은 인터페이스와 그 인터페이스를 통해서 시스템과 대화하는 상호작용(interaction)에 의해 정보가 발생한다. 따라서 TELOR는 최종 사용자와 시스템간의 상호작용 과정을 통하여 발생된 인터페이스 정보를 자연스럽게 분류 가능한 것이다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 관련 연구에 대하여 살펴보고, 3장에서는 역공학을 이용한 TELOR의 구조에 대하여 설명한다. 4장에서는 TELOR를 이용한 레거시 애플리케이션으로부터 인터페이스 객체를 추출하여 분류하는 과정의 사례를 보이며, 5장에서는 관련연구와 비교 및 의의를, 마지막으로 6장에서는 결론 및 향후 연구과제에 대하여 설명한다.

2. 관련 연구

본 장에서는 레거시 시스템 인터페이스로부터 객체 정보를 추출하는데 이용되고 있는 역공학과 관련된 내용을 기술한다. 역공학은 물리적인 환경으로부터 보다 추상적인 상위개념의 요소들을 복구하기 위한 역방향 프로세스이다. 이는 개발된 시스템으로부터 역으로 분석 및 설계 단계인 초기과정으로의 흐름이다. 표 1은 여러 가지의 역공학 방법론 중에서 몇 가지 종류의 특징들을 비교하여 그 장·단점을 파악하고 이를 본 연구를 수행하는데 적용하기위한 목적으로 비교 조사한 것이다.

2.1 데이타베이스 스키마를 입력 자료로 하는 역공학의 예

데이타베이스 역공학에 관한 예는 관계형 데이타베이스 스키마를 ER 다이어그램으로 변환하는 방법, 네트워크 스키마를 ER 다이어그램으로 변환하는 방법, 계층형 스키마를 ER 다이어그램으로 변환하는 방법, 객체지향 데이타베이스 스키마를 바이너리 관계 다이어그램으로 변환하는 방법 등 여러 가지가 있다[14]. 본 절에서는 현재까지도 레거시 시스템 에서 가장 널리 사용되고 있는 관계형 데이타베이스 스키마를 ER 다이어그램으로 역공학 하는 과정에 대하여 기술한다. 관계형 데이타베이스 스키마를 ER 다이어그램으로·역공학 하는 방법은 함수적 종속성(functional dependencies)과 내포 종속성(inclusion dependencies)을 기반으로 한다[14,15]. 또한 DBMS(Database Management System)내의 카탈로그(catalog) 정보와 기존 데이타베이스의 튜플(tuples)을 이용하여 데이타베이스의 개념적 다이어그램을 유도해낸다. 관계(relations)는 기본 키(primary key)와 외래키(foreign key)를 이용하여 정의된다. 이러한 관계 정의는 사용자와의 상호작용을 통해서 이루어진다. 그런 다음에는 관계의 정의를 통해서 관계형 스키마를 해석

표 1 역공학 방법론의 종류 비교

구분 / 내용	입력 자료	구분 관점	목표 모델	특징	비고
역공학 종류	DB 스키마	데이타	ER 다이어그램	· 키 기반 · 내포(inclusion) 종속성 기반 · 내포, 키 종속성의 결합	Batini[6], Navathe[7] Chiang[8], Shivak[9] 등의 방법론 사용
	소스 코드	프로세스	재사용 모듈	· 후보 기준(candidature criterion) 사용	RE2[10], Saleh[11] 방법론 사용
	양식 구조	객체	객체지향 다이어그램	· 애플리케이션 시스템 양식사용 · 객체 구조 및 시나리오	FORE 방법론 사용[12]
	소스 코드	객체	CTF(Compact Trace Format)	· 객체 상호작용 동작을 CTF 스키마를 이용하여 표현	Abdelwahab[13]
	인터페이스 구조 및 최종사용자와의 상호작용 정보	객체	객체지향 다이어그램	· 레거시 시스템의 인터페이스 사용 · 객체 구조 파악 · 객체 단위 분류 및 복구	본 연구

한다. 카디날러티(cardinality) 제약사항(constraints)은 함수적 종속성의 분석을 통해서 유도한다[6,7].

2.2 소스 프로그램을 입력 자료로 하는 역공학의 예

소스 프로그램을 입력 자료로 하는 역공학의 예는 소스 코드의 모듈 구조나 프로그램 논리 구조를 유도하는 것이 목적이다. 이를 위해 소스 프로그램을 분석하고 이를 다시 의미 있는 정보 단위로 분할하는 것이다. 대표적인 예로 RE2(Reuse Reengineering) 프로젝트가 있다[10]. RE2 연구 프로젝트는 CNR(Italian National Research Council)에 의해 투자되었으며, Naples 대학교의 DIS(Departments of 'Informatica e Sistemistica')와 Durham 대학교의 CSM(Research Institute in Software Evolution)에서 공동으로 수행되었다. RE2 프로젝트는 역공학과 재공학 기법들이 기존 소스 코드를 포함하는 재사용 재공학 프로세스에서 가지는 규칙의 밑그림을 그린다. 즉 소프트웨어 재사용을 목적으로 하고 있으며, 후보 기준(candidature criterion)을 설정하여 활용한다. 또한, 기존 시스템으로부터 재사용 가능한 모듈을 생성해 내는 것이 목적이다. 이를 위해서는 데이타 추상화를 시켜야 되는데 처리규칙에 기반한 기준이 제안되었으나 기준을 설정하는 것이 모호하다.

Saleh 방법론은 Saleh, Boujarwah에 의해 제안된 역공학 방법론이다. 이 방법은 Estelle라는 ISO에서 제정한 일반적으로 분산 시스템과 프로토콜을 위한 명세 언어로 통신 소프트웨어의 고 수준의 추상화를 얻는 것이 목적이다. 그러나 이 방법은 특정 소프트웨어 즉, 통신 소프트웨어에 종속적이다[11].

FORE 방법론은 양식 구조와 사용자 상호작용 정보를 입력 자료로 해서 객체구조를 생성해내는 부분은 본 연구와 비슷하나 객체 모델을 표현하기 위해서 CRC (Class Responsibility Collaboration) 카드 기법을 사용하고 있다. 이는 현재 거의 표준화 되어있는 UML (Unified Modeling Language)이라는 통합 모델링 언어가 이용되고 있고, 현재 개발되는 객체지향 시스템 또한 대부분 UML 표기법을 따르고 있으므로 추출된 객체 모델 정보가 현 상황에 맞지 않아 이러한 객체 정보를 현재의 시스템에 맞게 재사용 되려면 또다시 UML로 매핑(mapping) 되어야 하는 복잡함이 있으며 그 과정에서 발생할 수 있는 정보 오류의 가능성이 내재한다.

2.3 관련 연구의 고찰

많은 역공학 방법론 중에 몇 가지 주요 방법론에 대하여 살펴보았다. 본 절에서는 앞서 알아보았던 기존 역공학 방법론의 미진한 부분과 모순이 되는 부분들을 요약해본다. 첫째, 기존 역공학 방법들은 표 1에서도 알 수 있듯이 크게 소스코드 위주의 프로세스 관점과 데이타베이스 스키마에서의 데이타 관점 중에서 어느 한 쪽만을 다루고 있다는 점이다[12,16]. 즉, 소스코드를 중심으로 하는 프로세스 역공학 방법론은 프로그램의 모듈 구조와 모듈들 간의 관계 등을 유도해 내는 것이 핵심이고, 데이타 구조나 데이타베이스 스키마를 중심으로 하는 역공학 방법론은 개념적인 데이타 모형만을 유도해 내는 것이 핵심이다[17-19]. 따라서 기능적인 면은 고려되지 않고 있다. 하나의 시스템을 완전히 개발되기 위해서는 프로세스 측면과 데이타 측면이 동시에 만족되어야 한다. 둘째, 현재까지 제안된 역공학 방법론은 사용자 중심에서 바라보는 시각보다는 방법론 자체의 획일적인 자동화에 더 초점을 맞추고 있다. 물론, 역공학 자체를 자동화하는 것도 중요한 요소라 볼 수 있겠으나 그보다 먼저 사용자 중심의 시각에서 접근하는 방법이 더 중요하다. 셋째, 기존 방법론들을 통해 유도되

었던 목표 모델이 방법론마다 다르다는 점이다. 즉, 레거시 시스템으로부터 유도된 모델 정보들이 표준 표기법으로 표기되어야할 필요성이 있다. FORE 방법론은 CRC 표기법을 응용하여 자체적으로 개발한 ECRC (Extended CRC)표기법을 사용하고 있으나 이 역시 표준화된 표기법은 아니다.

본 논문에서 제안하는 TELOR는 애플리케이션 시스템의 인터페이스 정보를 입력 자료로 활용하는 것으로 사용자가 가장 먼저 접하고 사용자의 행위를 전달해줄 수 있는 인터페이스 화면을 이용한다. 물론 앞서 알아보았던 방법론에서 단점으로 제기된 데이타와 프로세스 두 가지를 동시에 추출하고자 한다. 이를 현재의 패러다임인 객체지향 시스템에 적용할 수 있도록 객체 단위로 추출하여 통합 모델을 유도해 내는 것이 TELOR의 목표이다. 객체단위로 유도된 의미 있는 정보는 레거시 시스템을 유지보수 하는데 용이할 뿐만 아니라 차세대 시스템을 개발하는데 개발 초기의 시스템 모델로 활용할 수 있다. 또한, 복구된 객체 모델을 이용하여 객체 모델 저장소가 구축이 된다면, 객체 정보가 저장소 내에 계속 축적이 되어 지속적으로 보완 발전될 수 있기 때문에 현 시스템의 유지보수 및 향후 연구내용에 유용하게 사용될 수 있다.

3. 객체추출 기법: TELOR

에이전트 기반 정보 수집기는 TELOR를 위해 지원될 램상주형 모니터링(감시) 프로그램의 한 종류이다. 이를 통해 수집된 정보는 인터페이스 지식 저장소에 저장이 되므로 다음 프로세스에서 이러한 정보를 호출하여 이용할 수 있다. 정보 수집기는 레거시 시스템이 어떠한 운영체제 상에서 실행되는가에 따라서 CUI 환경과 GUI 환경에서 운영되는 두 가지 형태로 분류될 수 있다. 다음은 이러한 두 가지 형태의 정보 수집기 프로토타입을 기술하고자 한다.

3.1 에이전트 기반 정보수집기의 정보 수집 예

레거시 시스템으로부터 객체 정보를 얻기 위한 과정은 레거시 시스템의 인터페이스 환경이 CUI와 GUI(본 논문에서는 마이크로소프트사의 운영체제를 표준 플랫폼으로 가정했음)인 형태가 있다. 본 논문에서는 CUI 환경의 레거시 시스템에 초점을 맞춰 객체추출 기법을 설명하고 있으나 GUI 환경에의 객체정보를 추출할 수 있는 정보수집기의 프로토타입 또한 설명하고자 한다. 먼저, CUI 환경에서는 VGA 그래픽카드의 경우 B000:B800은 텍스트 디스플레이를 위한 비디오 프레임 버퍼 공간이다. 따라서 CUI에서 실행되는 에이전트를 램상주 프로그램으로 작성하고, 특정 이벤트(일반적으로, 단축키를 누르면) 발생시 프레임버퍼를 스캔한다. 이렇게 하면 화면 정보를 그대로 불러올 수 있다. B000:B800 메모리는 <속성, 문자코드> 의 2바이트 정보가 반복되어 있기 때문에, 속성은 배경색, 전경색, 강조, 깜빡임 등의 속성이고, 문자코드는 씌어지는 문자의 코드 값이다. 예를 들어 화면에 "date:[ ]"라는 형식으로 되어 있다면 date를 변수로 잡을 수 있으며, [ ] 같은 문자나 색상이 달라지는 것을 통해서 입력을 위한 공간을 예측할 수 있다. 그러나 이러한 방법은 다음과 같은 단점이 있다. VGA 그래픽카드에서 텍스트 프레임 버퍼공간은 B000:B800이지만, 다른 종류의 카드나 모드에 따라서 달라질 수 있다. 따라서 하드웨어 특성에 종속적이지 않은 방법이 필요한데, BIOS의 10h 인터럽트는 주로 그래픽카드 관련된 표준을 지원한다. 10h 인터럽트에 02h 서비스(AX=02h)는 커서의 위치(x, y)를 설정할 수 있다. 그리고 위치의 속성과 코드 정보를 얻고자 할 때는 10h 인터럽트의 08h 서비스를 활용한다. 이렇게 하면 거의 모든 종류의 그래픽카드(Monochrome, CGA, EGA, MGA, VGA, XGA 등)를 일관성 있게 사용할 수 있는 장점이 있다.

둘째, GUI 환경에서는 GUI에 등록되어있는 객체 정보를 얻어낼 수 있는 소프트웨어(마이크로소프트사의 SPY++와 같은)를 이용하여 인터페이스 정보를 얻어낼 수 있다. 마이크로소프트사의 윈도우 환경이라면 SPY++라는 소프트웨어를 이용하여 인터페이스 객체정보를 획득할 수 있으므로 에이전트는 SPY++를 통해 획득된 정보를 토대로 객체정보를 추출한다. 다음은 SPY++를 실행시켜 GUI 환경의 레거시 시스템으로부터 획득한 정보를 추출하는 과정을 보여준다. 그림 1은 GUI 환경에서 실행되는 애플리케이션의 한 종류이다. 여기에서는 "Serial Number Generation and Add"라는 버튼에 해당되는 정보를 나타내주고 있다.

그림 2는 애플리케이션의 인터페이스에 나타나있는 객체정보(그림 2에서는 버튼 객체만을 예로 하였음)의 등록 사항을 나타내주는 그림이다. 그림에서 보는 바와 같이 각 버튼 객체와 인터페이스의 구성들도 알기 쉽게 등록되어있다는 것을 한눈에 볼 수 있다. 에이전트는 이러한 SPY++의 객체정보를 텍스트 파일로 변환한 다음 이를 인터페이스 저장소에 저장하여 다음 단계인 인터페이스 객체 분할 단계에서 이용된다.

3.2 TELOR의 객체추출 기법

TELOR는 4단계의 세부 프로세스 즉, 인터페이스 사용 사례 분석 단계, 인터페이스 객체 분할 단계, 객체구조 모델링 단계, 모델 통합 단계 등으로 구성되어 있다. 그림 3은 TELOR의 구조도를 도식화한 것이다.

첫 번째 인터페이스 사용 사례 분석 단계는 레거시 애플리케이션의 인터페이스 필드 정보와 사용자가 인터

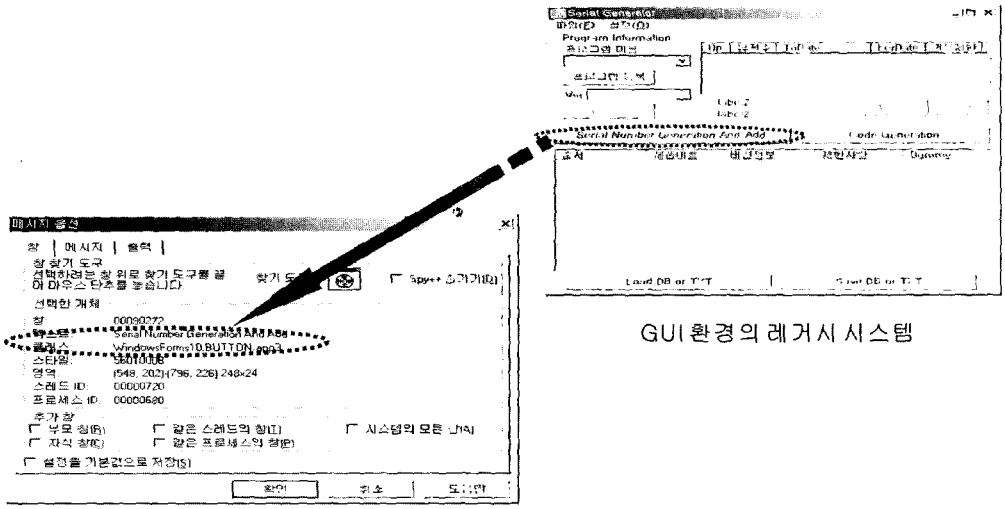

그림 1 애플리케이션의 객체정보 파악

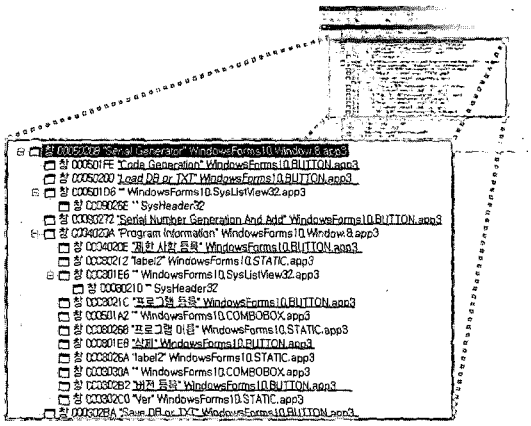

그림 2 SPY++를 통한 애플리케이션의 객체 등록 형태

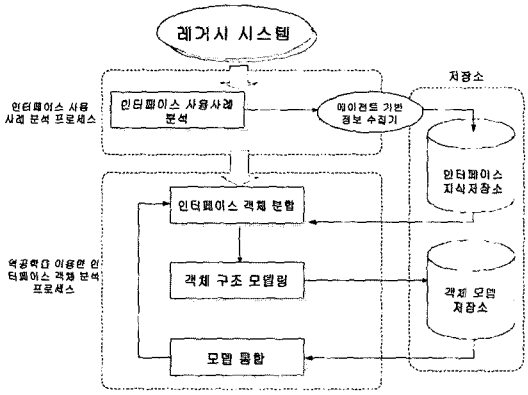

그림 3 TELOR의 구조도

페이스를 통해 시스템과 상호 작용한 정보를 얻기 위한 단계이다. 사용자가 레거시 시스템의 인터페이스를 통한 상호작용의 예는 입력 필드에 어떠한 데이타를 입력하는 행위, 입력한 데이타를 처리하기 위해 처리 시스템에 보내기 위한 이벤트 발생 행위, 이벤트에 의해 발생되어진 입력 데이타를 처리하는 연산행위, 연산행위를 통해 처리된 데이타를 사용자 인터페이스에 다시 반환하는 행위 데이타베이스로부터 데이타를 인터페이스에 넘겨주는 행위 등 여러 가지의 예가 있다. 그림 4는 인터페이스 사용 사례를 분석하기 위한 알고리즘을 나타낸다. 이러한 상호작용 정보와 인터페이스에 관한 정보가 에이전트 기반 정보 수집기에 의해 자동 수집된다.

에이전트 기반 정보 수집기에 의해 얻어질 수 있는 인터페이스 구성 정보유형은 다음 네 가지의 유형으로 분류될 수 있는데 인터페이스 영역유형, 입력지원 컨트롤, 인터페이스 편집필드유형, 인터페이스 처리유형 등이 있다.

첫째, 인터페이스 영역유형은 인터페이스의 영역별 기능을 나타내는 유형이다. 따라서 데이타의 항목을 나타낼 수 있는 항목별 필드 영역부분과 항목의 수를 카운트 할 수 있는 집계 필드 영역 또는, 그리드(grid) 형태로 나타낼 수 있는 그리드 영역, 그리고 이벤트를 컨트롤 할 수 있는 이벤트 컨트롤 영역, 마지막으로 조건에 따른 입력을 허용하는 조건부 입력 영역 등으로 세부 분류될 수 있다.

그림 5와 그림 6은 인터페이스를 구성하는 유형과 영역 유형 분류 및 적용 예를 나타내고 있다. 특히, 그림

```
DB  DBList, Sgl_DBList; // DBList는 모든 Sgl_DBList들에 대한 정보를 갖고 있는 연결리스트
while(!All_Interface_exit()) // 모든 인터페이스에 대하여 반복
{
   getSgl_Interface(); // 단일 인터페이스 획득
   while(!exit)
   {
      getInter_User_System(); // 에이전트 시스템을 통한 사용자와 시스템간의 상호작용 정보 획득
      Sgl_DBList = saveAll_Interface_Inform(); // 획득된 단일 인터페이스 정보를 저장
   } // end while
   addList(DBList, Sgl_DBList); // 단일 DBList를 DBList에 추가
} // end while
```

그림 4 인터페이스 사용 사례 분석 알고리즘

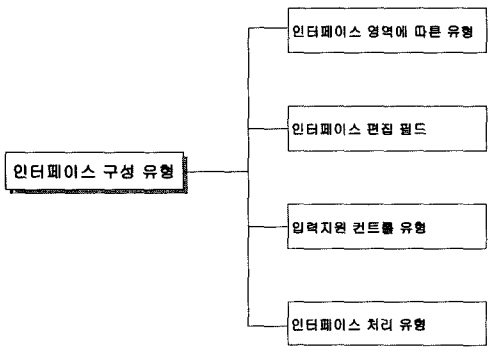

그림 5 인터페이스 구성 유형

영 역	① 항목별 필드 영역
	② 집계 필드 영역
	③ 그리드 필드 영역
	④ 이벤트 컨트롤 영역
	⑤ 조건부 입력영역

조 회	①②③④⑤
등록, 수정	①③④
삭 제	①③④⑤

그림 6 영역 유형 분류 및 처리 적용 예

6은 하나의 인터페이스에서 나타날 수 있는 처리 적용 예(조회, 등록, 수정, 삭제)를 나타낸 그림이다. 조회의 경우에는 사용자로 하여금 검색대상을 인터페이스로 나타날 수 있는 정보로서 검색대상을 항목별, 집계별, 그리드별로 나타낼 수 있는 경우가 있을 수 있고 시스템에 명령을 전달하는 이벤트 발생 역할을 하는 이벤트 컨트롤 영역과 조회하고자하는 특정 영역에 조건을 주기 위한 조건부 입력 영역이 나타날 수 있다. 등록, 수정의 경우에는 집계를 하는 부분과 조건을 두기 위한 부분이 불필요하므로 집계 필드 영역과 조건부 입력 영역 부분이 제외되었다. 그러나 삭제 시에는 조건부로 삭제가 필요하므로 집계 필드를 제외한 나머지 부분이 모두 쓰인다.

둘째, 입력지원 컨트롤 유형은 사용자가 인터페이스에 데이타를 입력하는 방법에 여러 유형이 있을 수 있는데 상황에 따라 사용자가 편리하게 데이타를 인터페이스에 입력할 수 있도록 한다. 그림 7은 인터페이스에 나타날 수 있는 입력지원 컨트롤의 종류를 나타낸다. 그림 7에서도 나타나있듯이 입력지원 컨트롤에는 콤보 박스, 서브 인터페이스, 라디오 버튼, 체크버튼, 관련항목 편집제공 등의 유형이 나타날 수 있다.

셋째, 인터페이스의 필드를 편집할 수 있는 인터페이스 편집필드가 있다. 그림 8에 나타나 있는 인터페이스 편집필드 유형은 사용자가 인터페이스 상에서 데이타를 시스템에 전달하기 위해 편집할 수 있는 필드의 유형을 구분한 것이다. 키 입력 필드와 데이타 입력필드, 조회와 수정 둘 모두 가능한 필드, 조회만 가능한 필드, 이벤트를 발생시킬 수 있는 필드, 마지막으로 시스템 관리를 위한 시스템 관리 필드의 유형이 존재할 수 있다.

넷째, 처리 적용형태에 따른 유형을 분류할 수 있다. 인터페이스 상에서 일어날 수 있는 처리 적용형태는 앞서 설명한 조회, 등록, 수정, 삭제의 크게 4가지가 있다. 이는 다시 인터페이스 상에서 이루어 질 수 있는 처리 건별로 단일건 처리와 다수건 처리로 구분될 수 있으며 이는 또다시 데이타베이스를 기준으로 해서 다른 테이블과의 조인 또는 무조인으로 구분될 수 있다. 지금까지는 인터페이스 사용 사례를 이용한 분석 프로세스에 대해 알아보았다. 나머지 세 단계는 역공학을 이용한 객체지향 개념을 모델링하는 과정이다.

TELOR의 두 번째 단계는 인터페이스 객체 분할 단계이다. 본 단계에서는 전 단계의 에이전트 프로그램에 의해 수집되었던 인터페이스에 관한 정보를 이용한다.

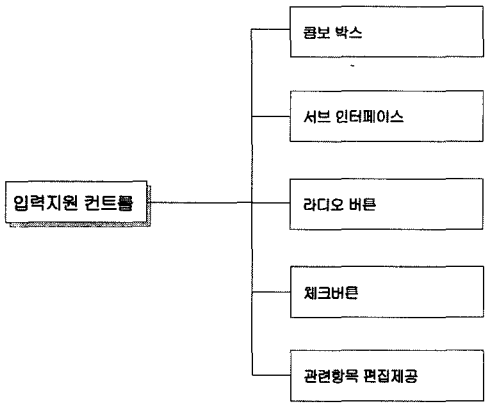

그림 7 입력지원 컨트롤 유형

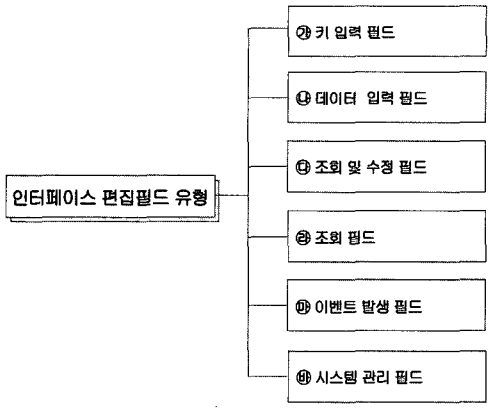

그림 8 인터페이스 편집필드 유형

인터페이스에 관한 정보가 저장되어있는 인터페이스 지식 저장소로부터 인터페이스에 관한 다양한 분류 정보를 입력받는다. 그런 다음 인터페이스에 나타나있는 다양한 필드들의 정보를 영역별(항목, 집계, 그리드, 이벤트, 조건부 영역 등) 및 입력지원 컨트롤(콤보박스, 서브 인터페이스, 라디오 버튼, 체크버튼 등) 그리고 인터페이스 편집필드 유형(키 입력 필드, 데이타 입력필드, 조회 및 수정 필드, 조회 필드, 이벤트 발생 필드, 시스템 관리필드) 등의 인터페이스 정보의 유형에 따라서 인터페이스에 관한 지식을 의미 있는 정보 단위 객체로 분할한다. 그림 9는 인터페이스의 객체 분할 절차를 나타내는 알고리즘이다.

표 2는 위와 같은 방법으로 분할된 인터페이스 정보를 처리 적용 예(조회, 수정, 등록, 삭제)별로 건별 처리 유형(단일건 처리, 다수건 처리)별로 데이타베이스 상에서의 테이블 간의 조인 여부(조인, 무조인)별로 그리고 사용 예에 따른 처리 유형별로, 인터페이스 편집필드 유형별로 구분한 도표이다. 특히, 처리 유형에 따른 분류표와 인터페이스 편집필드 유형 분류 표 내의 원안 숫자와 원안 문자는 사용 예에 따른 그림 6 영역 유형 분류와 처리 적용 예 그리고 그림 8의 인터페이스 편집필드 유형에서 구분한 분류표를 기준으로 작성하였다.

표 2 처리 적용 형태별 처리유형 및 편집필드 구성 표

처리 적용 예	건별 처리유형	조인 여부	사용 예	처리유형에 따른 분류	폼 편집필드 유형
조회	단일건 처리	무조인 처리	편집, 조회	①, ④	㉮ ㉱ ㉲ ㉳
			조회	①, ④	㉮ ㉱ ㉲ ㉳
		조인 처리	편집, 조회	①, ②, ④	㉮ ㉱ ㉲ ㉳
			조회	①, ②, ④	㉮ ㉱ ㉲ ㉳
	다수건 처리	무조인 처리	편집, 조회	①, ③, ④	㉮ ㉱ ㉲ ㉳
			편집, 조회, 집계	①, ②, ③, ④	㉮ ㉱ ㉲ ㉳
			조회	①, ③, ④	㉮ ㉱ ㉲ ㉳
		조인 처리	편집, 조회	①, ③, ④	㉮ ㉱ ㉲ ㉳
			편집, 조회, 집계	①, ②, ③, ④	㉮ ㉱ ㉲ ㉳
			조회	①, ③, ④	㉮ ㉱ ㉲ ㉳
수정	단일건 처리	무조인 처리	등록	①, ④	㉮ ㉯ ㉱ ㉲ ㉳
		조인 처리	편집	①, ④	㉮ ㉯ ㉱ ㉲ ㉳
			등록	①, ④	㉮ ㉯ ㉱ ㉲ ㉳
	다수건 처리	무조인 처리	등록	③, ④	㉮ ㉯ ㉱ ㉲ ㉳
			편집, 등록	③, ④	㉮ ㉯ ㉱ ㉲ ㉳
		조인 처리	등록	③, ④	㉮ ㉯ ㉱ ㉲ ㉳
			편집, 등록	③, ④	㉮ ㉯ ㉱ ㉲ ㉳
등록	단일건 처리	무조인 처리	등록	①, ④	㉮ ㉰ ㉱ ㉲ ㉳
		조인 처리	편집, 등록	①, ④	㉮ ㉰ ㉱ ㉲ ㉳
삭제	단일건 처리			①, ④, ⑤	㉮ ㉱ ㉲ ㉳
	다수건 처리			①, ③, ⑤	㉮ ㉱ ㉲ ㉳

```
while (DBList != NULL) // DBList: 에이전트에 의해 획득된 모든 정보 리스트
{
  Object  InterOb, Sgl_InterOb; // 인터페이스 객체
  Object  ComplexOb, Sgl_ComplexOb; // 복합 객체
  Object  DBaccessOb, Sgl_DBaccessOb; // DB 접근 객체
  Object  ExtraOb, Sgl_ExtraOb; // 부가적인 객체

  Sgl_InterOb = createInterfaceObject(DBList); // 단일 인터페이스 객체 생성
  giveNameToObject(Sgl_InterOb.name); // 인터페이스 이름을 이용하여 객체명 부여
  givePropertyToField(Sgl_InterOb.field); // 인터페이스 필드에 속성 할당
  addList(InterOb, Sgl_InterOb); // 인터페이스 객체 리스트에 추가

  while (Sgl_InterOb.field != NULL) // 단일 인터페이스내의 모든 필드 검색
  {

    if (isComplexField(Sgl_InterOb.field)) // 복합 필드 검색
    {
      Sgl_ComplexOb = createObject(Sgl_InterOb.field); // 복합 필드를 복합 객체로 유도
      giveNameToObject(Sgl_ComplexOb.name); // 객체명 부여
      givePropertyToField(Sgl_ComplexOb); // 속성 부여
      addList(ComplexOb, Sgl_ComplexOb); // 객체 리스트에 추가
    } // end if

    if (isDBaccess(Sgl_InterOb.field)) // DB 접근 필드 검색
    {
      Sgl_DBaccessOb = createObject(Sgl_InterOb.field); // DB 접근 필드를 DB 접근 객체로 유도
      giveNameToObject(Sgl_DBaccessOb.name); // 객체명 부여
      givePropertyToField(Sgl_DBaccessOb.field); // 속성 부여
      addList(DBaccessOb, Sgl_DBaccessOb); // 객체 리스트에 추가
    } // end if
    else if (checkObject(Sgl_InterOb.field)) // 부가적인 필드 검색
    {
        Sgl_ExtraOb = createObject(Sgl_InterOb.field); // 부가적인 필드를 부가적인 객체로 유도
        giveNameToObject(Sgl_ExtraOb.name); // 객체명 부여
        givePropertyToField(Sgl_ExtraOb); // 속성 부여
        addList(ExtraOb, Sgl_ExtraOb); // 객체 리스트에 추가
    } // end if
    Sgl_InterOb.field = Sgl_InterOb.field.next; // 다음 필드로 이동
  } // end while
  DBList = DBList.next; // 다음 단일 인터페이스로 이동
} // end while
```

그림 9 인터페이스 객체 분할 알고리즘

TELOR의 세 번째 단계는 객체 구조 모델링 단계이다. 본 단계에서는 이전 과정의 단계에서 나타난 결과로부터 객체들을 파악하는 단계이다. 또한 객체들 간의 초기 수준의 구조적(structural)관계와 협력(collaboration) 관계가 유도되는 단계이다. 본 단계에서 나타나는 결과물은 유도된 객체의 객체 명, 객체 속성, 그리고 구조적 관계로 구성된 객체구조 모델이 형성된다. 또한 객체구조 모델을 나태내기 위해 사용한 표기법으로는 현재 모델링 언어의 표준인 UML을 이용하여 나타낸다. 그림 10은 객체 구조를 모델링하기 위한 절차를 나타내는 알고리즘이다.

TELOR의 마지막 단계인 네 번째 단계는 모델 통합 단계이다. 모델 통합 단계의 목적은 전 단계의 결과물인 객체정보를 나타내는 단위 모델들을 통합하여 보다 상위 수준의 통합 모델을 제시하고자 함이다. 결과적으로, TELOR의 과정을 통해 얻어지는 최종 결과물은 객체 구조 모델이다. 그림 11은 모델 통합을 위한 알고리즘을 나타낸다.

지금까지 레거시 시스템의 인터페이스를 기반 하여 역공학을 이용한 TELOR의 구조와 각 단계별 세부 내용에 대해 알아보았으며, 이를 실제 레거시 시스템을 이용하여 본 프로세스를 적용시키기 위해 필요한 전처리

```
while (InterOb != NULL) // 모든 인터페이스 검색
{
  while (InterOb.ExtraOb != NULL) // 모든 집계 필드 후보 검색
  {
    if (isCompute_field(InterOb.ExtraOb)) // 집계 필드 검증
    {
      Compute_Inform = callAgentSystem(InterOb.ExtraOb);
          // 에이전트로 부터 집계 필드와 관계된 모든 정보 획득
      while (Compute_Inform != NULL)
      {
        operator = searchOperator(Compute_Inform); // 최소단위 연산자 검색
        operand = searchOperand(Compute_Inform); // 최소단위 피연산자 검색
        variable = initVariable(operand); // 변수 설정
        relationOb = searchObject(InterOb, operand); // 피연산자와 관계된 객체 검색
        setRelation(variable, relationOb); // 객체와 변수와의 관계 설정
        method = createMethod(variable, operator); // 연산자와 피연산자와의 계산식 생성
        Compute_Inform = Compute_Inform.next; // 집계 필드에 관계된 다음 정보로 이동
      } // end while
      setMethodToObject(method, InterOb.ExtraOb); // 집계 필드 객체에 메소드 할당
      InterOb.ExtraOb = Sgl_ExtraOb.next; // 다음 집계 필드 후보로 이동
    } // end if
  } // end while
  InformOb = InformOb.next; // 다음 인터페이스로 이동
} // end while
```

그림 10 객체 구조 모델링 알고리즘

```
model = NULL; // model 변수의 초기값 설정
alter_model = NULL; // alter_model 변수의 초기값 설정
while (!isSuitableModel(model)) // 모델 적합성 판단
{
  model = selectStruc(InterOb); // 전체 인터페이스 객체 구조 모델 선택
  InterOb = InterOb.next; // 다음 인터페이스로 이동
  while (InterOb != NULL) // 모든 인터페이스 객체 검색
  {
    alter_model = selectStruc(InterOb); // 이동한 인터페이스의 객체 구조 모델 선택
    if (isDifferModel(model, alter_model)) // 두 모델의 유사성 판단
    {
      mergeSameOb(model, alter_model); // 공통 객체를 기반으로 모델 통합
      mergeByRelation(model, alter_model); // 관계있는 모델 통합
      solveAmbiguousName(model, alter_model); // 모호한 이름 해결
      solveAmbiguousStruc(model, alter_model); // 모호한 구조 해결
      model = saveToAll(); // 통합된 모델 저장
    } // end if
    InterOb = InterOb.next; // 다음 인터페이스 지정
  } // end while
} // end while
```

그림 11 모델 통합 알고리즘

단계가 필요하다. 전처리 단계는 본 TELOR에 속하지는 않지만 레거시 시스템의 역공학적 객체 분석을 하기 위해 설정되어야 한다. 첫 번째 전처리 단계는 객체분석 대상 레거시 시스템을 선정하는 것이다. 본 단계에서는 대상 범위와 그에 관련된 정보를 정하고, 레거시 시스템의 주요 사용자를 파악하는 단계이다. 두 번째 전처리 단계는 레거시 시스템에서 이용하는 인터페이스의 영역이 정의되어야 한다. 하나의 인터페이스를 처리하는 하나의 태스크(task)를 인터페이스 영역으로 가정하기로 한다. 인터페이스 영역 간에 종속성이 파악되어야 한다. 즉, 어떠한 인터페이스를 먼저 처리하고 그 다음에 어떠한 양식을 처리해야 하는 지의 선후 관계가 명확히 정해져야 한다.

4. TELOR 적용 사례

4.1 사례

본 논문에서 제안한 TELOR를 이용하여 레거시 시스템의 인터페이스 정보로부터 객체모델을 획득하는 과정을 실 예로 보이기 위한 과정을 본 장에서 기술한다. 실 예를 보이기 위해 선택한 레거시 시스템으로는 유통현

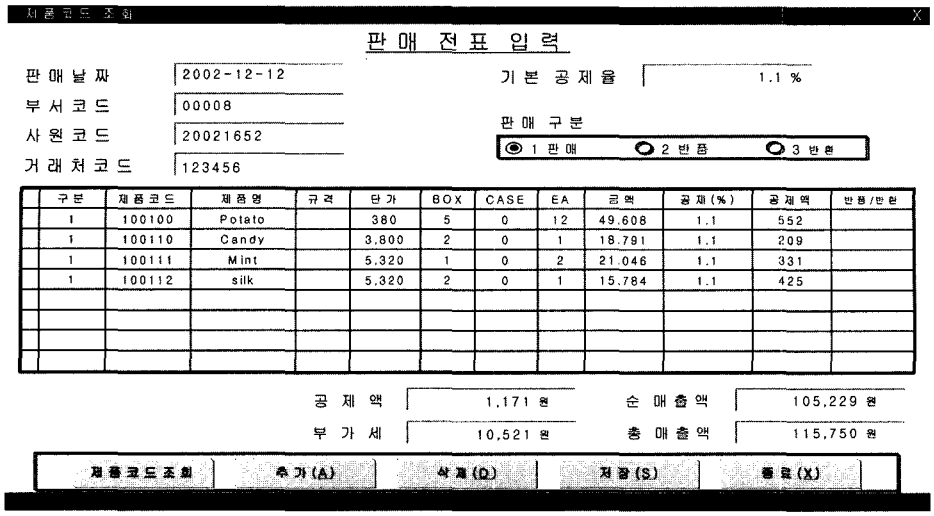

그림 12 판매전표 관리 인터페이스

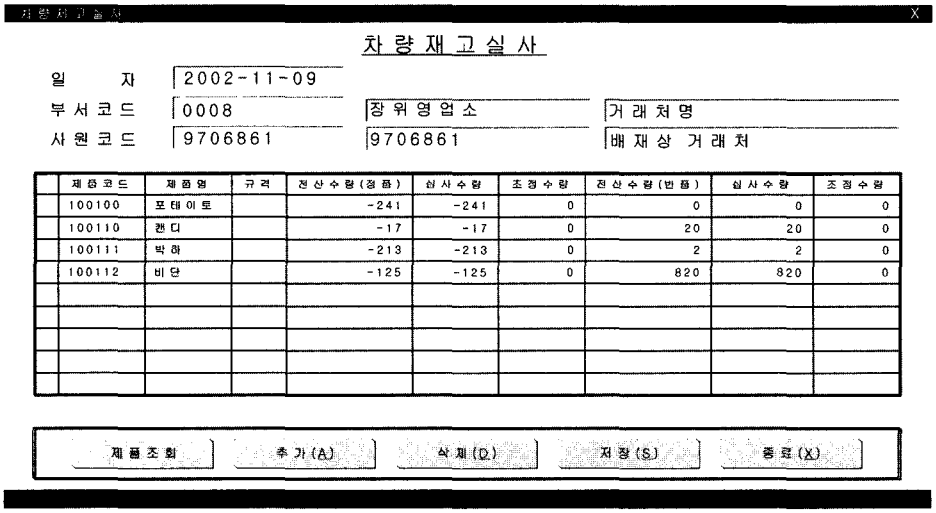

그림 13 차량재고 실사 인터페이스

황을 다루는 기업에서 사용하고 있는 SFA(Sales Force Automation) 시스템을 예로 한다. SFA 시스템은 기업의 제품 판매를 강화하기 위한 전산화 시스템으로 유통망을 가진 대부분의 기업에서 사용하는 레거시 시스템이다. 그림 12와 그림 13은 SFA 시스템 인터페이스의 일부이다. TELOR가 객체구조 모델을 획득하는 것이 목적이므로 인터페이스의 구조뿐만 아니라 이벤트 발생에 의한 처리 결과를 나타내기 위한 비즈니스 로직 부분도 필요하다. 따라서 인터페이스 외에 사용자가 인터페이스를 사용한 업무 처리 절차의 사용 사례를 다음과 같이 가정하기로 한다.

(1) 판매전표 관리 인터페이스 사용 사례

- 사용자인 영업사원은 판매를 마치고 판매전표 관리 인터페이스를 보이기에 앞서 시스템은 사용자 ID와 패스워드를 요구한다.
- 사용자는 ID와 비밀번호를 입력한다.
  (사용자가 영업사원임을 시스템이 인식하면 판매전표 입력 인터페이스를 사용자에게 제공함)
- 만약 사용자 ID와 패스워드가 잘못입력이 되었으면 세 번까지 허용하고 그 이상이면 시스템을 종료한다.
- 판매전표 입력 인터페이스가 나타나면 판매일자는 자동적으로 시스템의 오늘 날짜가 자동으로 입력된

다. (부서코드, 사원코드, 거래처코드는 사원이 직접 입력한다. 그러면 자동으로 입력상태가 된다)
- 영업사원이 판매했던 내역을 제품코드를 이용하여 입력한다. (제품코드는 아래의 제품코드조회라는 버튼을 누르면 데이타베이스로부터 자료를 가져와 자동입력이 됨)
- 그리드 창에 입력이 완료되면 저장 버튼을 클릭하여 저장한다. (나머지 항목은 자동계산 됨)

(2) 차량재고 실사 인터페이스 사용 사례
- 차량재고 실사 인터페이스의 메뉴를 선택하면 차량재고실사를 입력하는 화면이 나타난다.
- 일자, 부서, 사원코드를 입력하고 조회버튼을 클릭하면 사용자(영업사원)의 판매정보가 데이타베이스로부터 그리드 영역에 나타난다. 실사수량을 입력하면, 조정수량이 자동으로 계산된다.
- 새로운 제품을 추가할 경우 추가버튼을 이용하여 제품코드와 실사수량을 입력하면 된다.
- 입력 작업을 마치고 저장 버튼과 종료 버튼을 클릭하여 입력을 완료한다.

4.2 인터페이스 사용 사례 분석 단계

본 단계에서는 인터페이스에 관한 상세 객체 정보를 얻는 단계이다. 레거시 애플리케이션(SFA 시스템)의 인터페이스는 주문, 세금, 제품 송장 및 공급 요청 등과 같이 다양한 종류의 정보를 처리하는 방법을 단순화하고 표준화하기 위해 가장 보편적으로 사용자가 사용하는 수단이다. 인터페이스의 연산은 시간에 따라 업무를 제어하며 인터페이스의 흐름은 업무에 있어서 데이타 흐름이 된다. 이와 같은 객체적인 특성 때문에 인터페이스를 가장 원시적인 형태의 객체로 다룬다.

인터페이스 사용 사례 분석단계는 인터페이스가 가지고 있는 이러한 특성들을 기초로 객체 구조와 메소드를 찾는 단계이다. 이 과정에서의 주요 획득 정보는 인터페이스에 관한 정보이다. 인터페이스에 관한 정보는 두 부분으로 구분할 수 있는데 인터페이스 구조 부분과 사용자 연산 및 데이타베이스 접근에 관한 것이다. 표 3은 SFA 시스템의 판매전표 입력에 대한 정보의 한 예로 3장의 그림 4 사용 사례 정보 획득 절차를 토대로 수집될 가상의 정보 중에서 인터페이스 구조에 관한 정보를 표로 작성한 것이다.

표 3의 인터페이스 구조에서 캡션 이름은 인터페이스 필드의 의미 정보를 쉽게 파악할 수 있도록 해주는 역할을 하는 필드이다. 레거시 시스템이 실행되는 동안 인터페이스 필드의 값이 변수 이름에 실질적으로 할당된다. 인터페이스 필드 유형은 앞서 3장에서 구분한 것처럼 크게 입력지원 컨트롤/인터페이스 필드 유형/영역별 유형/이벤트별 유형 등으로 구분되므로 각 필드의 변수가 해당되는 필드 유형이 표 3의 오른쪽 항에 나타나있다. 시스템 필드는 시스템에 의해 컨트롤되는 부분인데, 날짜와 같이 시스템으로부터 얻어오는 정보가 여기에 해당된다. 데이타 입력 필드는 사용자가 단순히 단일의

표 3 판매전표 입력 인터페이스에 관한 정보 예 : 인터페이스 구조

내 용			입력지원 컨트롤/인터페이스 필드 유형 /영역/이벤트별 유형
구분	캡션 이름	변수 이름	
인터페이스 이름	판매전표 입력	pan_FSSAT01_v	인터페이스 객체
인터페이스 필드	판매일자	date_v	시스템 필드
	부서코드	dept_code_v	데이타입력 필드
	사원코드	emple_code_v	데이타입력 필드
	거래처코드	custom_code_v	데이타입력 필드
	기본 공제율	deduct_rate_v	데이타입력 필드
	판매구분	pan_v, back1_v, back2_v	라디오 버튼
	구분, 제품코드, 제품명, 단가, BOX, CASE, 금액, 공제, 공제액	divide_v1, divide_v2, … p_code_v1, p_code_v2,… p_name_v1, p_code_v2, …	그리드 필드
	공제액	d_amount	집계 필드
	부가세	a_amount	집계 필드
	순 매출액	p_amount	집계 필드
	총 매출액	s_amount	집계 필드
	제품코드조회	p_code_inq_v	이벤트발생 필드
	추가	p_add	이벤트발생 필드
	삭제	p_del	이벤트발생 필드
	저장	p_sav	이벤트발생 필드
	종료	p_exit	이벤트발생 필드

표 4 판매전표 입력 인터페이스에 관한 정보 예 : 인터페이스 동작 순서 및 결과

동작순서	동작	상응 인터페이스 필드	결과 내용
1	시스템제공 연산	date_v	2002-12-12
2	사용자 데이타 입력	dept_code_v	00008
3	사용자 데이타 입력	emple_code_v	20021652
4	사용자 데이타 입력	custom_code_v	123456
5	사용자 데이타 입력	deduct_rate_v	1.1%
6	라디오버튼 입력	rad_pan_v	rad_pan_v = true
		rad_back1_v	
		rad_back2_v	
7	액션 이벤트 (제품코드조회 버튼)	p_code_inq_v	p_code_inq_v = true
8	데이타베이스 접근	divide_v1, divide_v2, …	1
		p_code_v1, p_code_v2, …	100100
		p_name_v1, p_name_v2, …	박하사탕
		p_value_v1, p_value_v2, …	5,320
9	집계	d_amount_v	1,171
10	집계	a_amount_v	10,521
11	집계	p_amount_v	105,229
12	집계	s_amount_v	115,750
⋮	⋮	⋮	⋮

데이타를 입력하는 더 이상 나눌 수 없는 원소 형태의 필드이다. 그리드 필드는 하나의 동일한 이름으로 같은 의미의 인터페이스 필드들이 배열처럼 반복적으로 연관되어 있는 그룹 필드 형식이다. 집계 필드는 값이 어떠한 연산에 의해서 계산되는 필드를 말한다. 인터페이스 연산에 관한 정보의 사용 예가 표 4에 제시되어 있다.

표 4에 나타나있는 각 항목들의 의미는 다음과 같다. 동작 순서는 사용자와 레거시 시스템간의 상호작용 순서를 의미한다. 연산을 위한 동작 형태는 시스템제공 연산, 사용자 데이타 입력, 액션 이벤트, 데이타베이스 접근, 집계 등으로 구분될 수 있다. 각 동작에 상응하는 인터페이스 필드가 바로 오른쪽 항목으로 나타나 있다. 즉, 상응 인터페이스 필드는 연산자(operator)에 의해 이용되는 변수이다. 오른쪽의 결과 내용 항목은 처리가 완료된 후 변수에 할당된 값을 의미한다.

4.3 인터페이스 객체 분할 단계

본 단계는 지식저장소에 저장된 인터페이스 정보에 관한 내용을 분석하고 이를 보다 의미 있는 정보 단위로 분할하기 위한 단계이다. 즉, 복잡한 인터페이스 단위의 인터페이스 객체를 보다 단순한 객체단위로 분할하는 것이 본 단계의 목적이다. 인터페이스 객체는 3장에서 분류했던 여러 요소들(인터페이스 영역, 입력지원 컨트롤, 인터페이스 편집필드, 인터페이스 처리유형 등)로 구성이 되어있다. 이러한 요소들이 객체후보들이다. 이렇게 확인된 객체들을 추출하기 위해 객체를 나타낼 수 있는 표기법으로는 현재 모델링 언어의 표준인 UML을 이용한다.

이 과정은 3장의 그림 9와 같은 절차를 갖는다. 첫 번째 단계는 판매전표 관리 인터페이스를 이용해 한 본 단위의 인터페이스 객체를 생성한다. 한 본 단위 인터페이스 객체는 판매전표 관리 인터페이스의 필드 모두를 속성으로 가지며 그림 14는 판매전표 입력 인터페이스(클래스 명: pan_FSSAT01)의 클래스 다이어그램을 나타내고 있다.

두 번째 단계는 판매전표 입력 인터페이스 객체 내의 속성 중에서 복합 필드(예, 그리드 필드, 입력지원 컨트롤)들을 별도로 추출해낸다. 즉, 복잡한 요소들로 구성되어있는 클래스를 평탄화를 시키는 단계이다. 이렇게 함으로써 복잡했던 커다란 객체로부터 작은 단위의 객체 분리를 유도해 낼 수 있다. 이렇게 유도된 작은 단위의 객체들에게도 객체 이름을 부여한다. 그림 15는 판매전표 입력 인터페이스 객체로부터 분리된 작은 단위의 객체(Pan_FSSAT01, Pan_FSSAT01_grid)) 들을 보여주고 있다. 판매전표 입력 인터페이스의 예에서는 그리드 필드가 여기에 해당되므로 그리드 필드 부분을 따로 분리하였다. 따라서, Pan_FSSAT01_grid 객체는 Pan_FSSAT01 객체로부터 새롭게 생성된 객체이다. Pan_FSSAT01 인터페이스 객체를 분리한 후 객체가 단순화되었음을 그림 14와 그림 15를 통해 알 수 있다. 따로 분리된 작은 단위의 객체 이름 명명은 본래의 객체이름과 구분하기 위하여 본래 객체 이름에 새로 추출된 객체 이름(즉, 인터페이스 필드 명)을 합성하여 명명하기

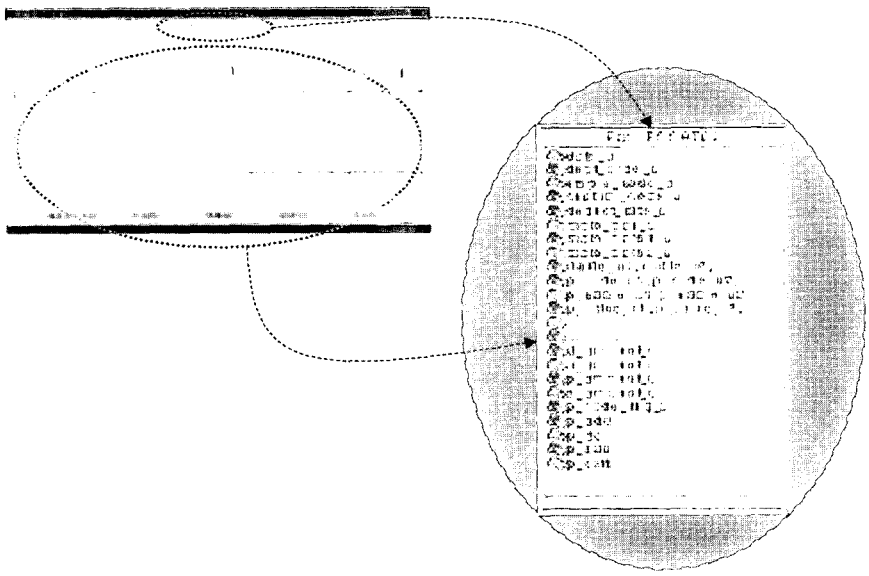

그림 14 판매전표 입력 인터페이스 객체

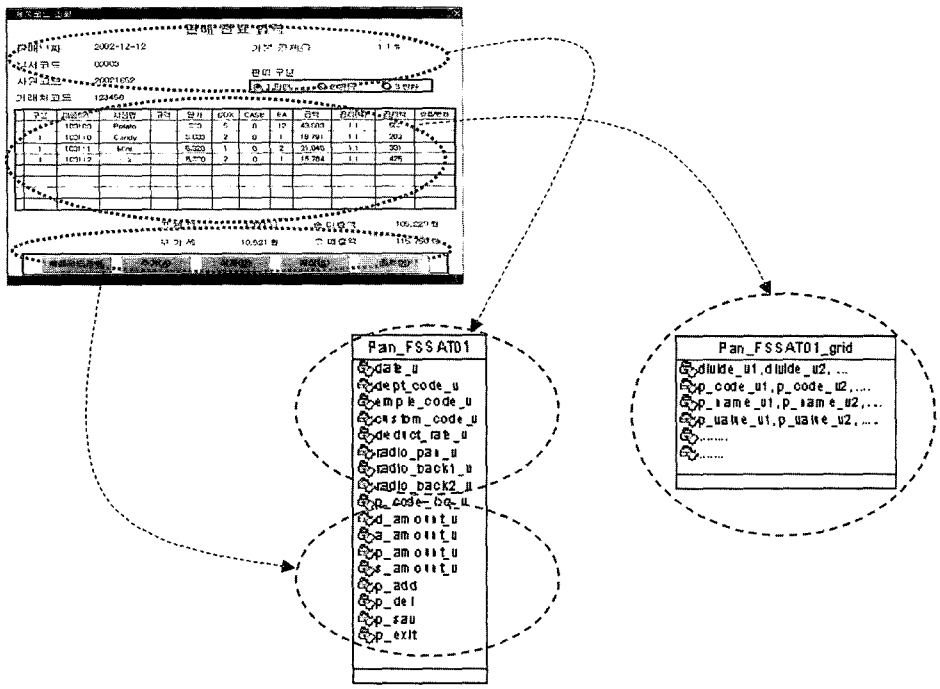

그림 15 Pan_FSSAT01, Pan_FSSAT01_grid로 평탄화된 판매전표 입력 인터페이스 객체

로 한다.

세 번째 단계는 각 객체내의 필드 중에서 어떠한 필드가 데이타베이스에 접근하는지를 파악하는 단계이다. 그림 15에서 데이타베이스에 접근하는 필드는 Pan_FSSAT01_grid 객체의 divide_v1, divide_v2, …, p_code_v1, p_code_v2, …, p_name_v1, p_name_v2, …, p_value_v1, p_valude_v2, … 등이 데이타베이스에 접근하는 필드 변수이다. 이들 값은 여러 데이타베이스 테이블 중에서 사원코드(CEMP01TT)테이블, 제품코드(CGDS01TT)테이블, 부서코드(CDEP01TT)테이블, 업

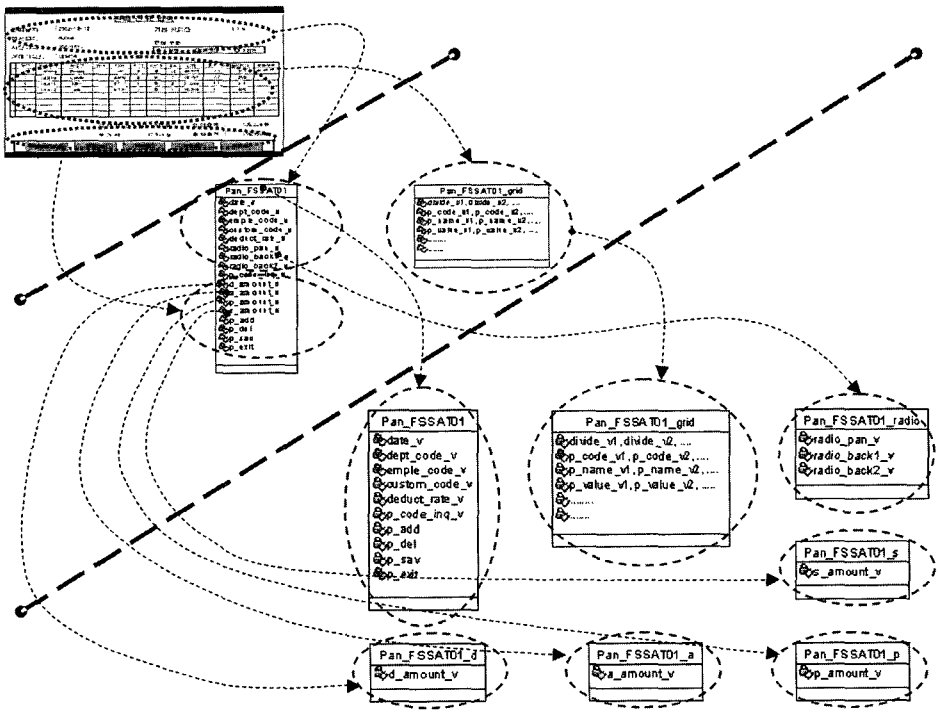

그림 16 판매전표 입력 인터페이스 객체에 대한 분리 및 부가적인 객체로 파악된 객체

체코드(CDCS01TT)테이블 등으로부터 가져오게 된다.

네 번째 단계는 부가적인 후보객체들을 추출하는 단계이다. 앞선 단계에서 발견되지 못했던 객체들을 최종적으로 확인하는 단계라 볼 수 있다. 그림 16은 판매전표 입력 인터페이스 객체로부터 분리된 작은 단위의 객체 및 추가로 파악된 객체를 나타내고 있다. 그림에서 데이타베이스에 접근하여 테이블로부터 데이타를 가져오는 필드인 Pan_FSSAT01_grid 객체의 속성들은 이들을 이용하여 공제액, 부가세, 순 매출액, 총 매출액을 계산하므로 부가적으로 Pan_FSSAT01_d, Pan_FSSAT01_a, Pan_FSSAT01_p, PanFSSAT01_s 등의 객체가 필요하다.

4.4 객체 구조 모델링 단계

객체 구조 모델링 단계는 전 단계에서 분리된 객체들을 이용해서 만들어진다. 전 단계에서 분리되었던 객체들은 속성은 파악이 되었지만 어떠한 행위(동작)를 하려면 메소드가 필요하나 메소드가 정의되어있지 않다. 본 단계에서는 객체의 연산 부분을 파악하고자 하는데 그 이유는 연산 부분이 바로 객체의 행위를 나타내므로 메소드의 주요 후보가 될 수 있기 때문이다. 이러한 연산 부분을 나타내고 있는 필드가 집계 필드이다.

집계필드는 최소단위 연산자와 단일 필드들로 분할된다. 메소드 정의는 집계 필드의 이름을 기반으로 해서 유도할 수 있다. 왜냐면, 변수 이름에 해당하는 집계 필드 명에는 처리 논리구조가 함축되어 있기 때문이다. 본 단계에서는 시스템에 대한 개발 경험이 풍부한 분석 전문가의 도움이 필요한 단계이기도 하다.

사용자 데이타 입력 필드와 데이타베이스 접근을 통한 값 배정 필드를 제외하면 대부분 집계(연산)필드라 볼 수 있다(물론 그 외에도 앞서 정의했던 입력 지원 컨트롤 등과 같은 필드도 존재하지만 여기서는 중요한 사항은 아니므로 무게를 두지 않기로 한다). 이러한 정보를 이용해서 집계 필드를 분석하여 객체의 메소드를 찾는다. 이러한 작업이 완료되면, 이를 이용해서 객체간의 관계도 밝혀낼 수 있다. 특히, 협력(collaboration) 관계는 연산에 관련된 필드들의 연관성으로부터 유도가 가능하다. 3장 그림 10 객체 구조 모델링 알고리즘에 나타나 있듯이 집계 필드 검색을 통해 Pan_FSSAT01 인터페이스 객체로부터 네 개의 집계 필드 즉, 공제액(d_amount_v), 부가세(a_amount_v), 순 매출액(p_amount_v), 총 매출액(s_amount_v) 등이 확인되었다.

그 다음 단계는 파악된 메소드의 할당 위치이다. 기본적으로 메소드는 주·종 관계에서 주(master)에 위치되어야 한다. 이러한 관계는 책임성(responsibility)을 살펴보면 쉽게 파악할 수 있다. 예를 들면, deduct_sum(), add_sum(), pure_sum(), total_sum() 등의 메소드는 판

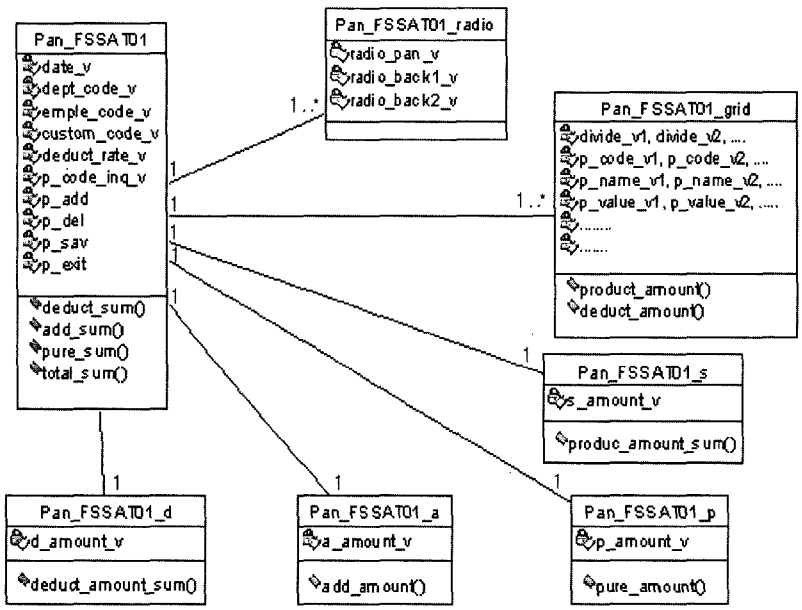

그림 17 판매전표 입력 인터페이스에 대한 객체 구조 모델

매전표 입력 인터페이스 즉, pan_FSSAT01_v에서 해당 연산을 수행해야 하므로 모두 pan_FSSAT01_v 객체의 책임 하에 있다고 볼 수 있다. 마지막으로 이렇게 파악된 주·종 관계의 객체 관계를 표현하기 위해 클래스 다이어그램으로 나타내면 된다. 그림 17은 클래스 다이어그램을 통한 판매전표 입력 객체와 그와 관련된 여러 객체들과의 관계를 보여주고 있다.

4.5 객체 모델 통합 단계

본 단계에서는 여러 인터페이스 객체로부터 유도된 객체 구조 모델이 하나의 모델로 통합되는 단계이다. 또한 결과로 생성된 통합 객체를 좀 더 완전한 객체로 만들기 위해 객체의 일반화/상세화(generalization/specialization)를 이용한 계층구조의 원리를 적용한다. 따라서 보다 추상화 된 상위의 모델을 얻을 수 있다. 본 단계의 목적은 여러 인터페이스 객체들로부터 생성된 모델을 기존 객체 모델과 통합하는 것이다. 본 예에서는 두 개의 인터페이스 객체를 이용했다. 판매전표 입력 인터페이스 객체 구조는 이미 앞선 단계에서 유도되었다. 또 다른 인터페이스 객체는 차량재고 실사라는 그림 13에 나타나있는 인터페이스이다. 차량제고 실사 인터페이스의 객체 구조는 그림 18과 같다.

그림 17과 그림 18의 객체 구조를 통해서 알 수 있는 것은 판매전표 입력 인터페이스의 Pan_FSSAT01 객체와 차량재고 실사 인터페이스의 Car_FSSAT09 객체와의 관계이다. 두 객체는 동일한 사원이 이용할 수 있는 속성들을 가지고 있다. 따라서 이를 대표할 수 있는 보다 추상화 된 객체가 존재할 수 있다는 점이다. 이와 같은 방법으로 두 객체가 통합된 모형이 그림 19에 나타나있다. Sup_Employee 객체로부터 Pan_FSSAT01와 Car_FSSAT09 객체가 상속을 받고 있다는 것을 알 수 있으며 두 객체와 관련된 객체들간의 연관 관계를 나타내고 있다.

통합된 객체 모델이 완성되면 일반화/상세화의 원리를 반복적으로 적용한다면 통합된 결과 모델은 점점 완전한 모습을 갖출 수 있다. 다만 통합 과정에서는 이름 및 구조의 충돌(conflict)현상이 발생할 수 있는데 본 논문에서는 새로이 유도되는 객체 이름을 본래의 객체 이름과 추출된 객체 또는 필드 이름을 결합하여 명명하는 특별한 명명 규칙을 사용하여 이름 충돌 현상을 방지할 수 있다.

5. 관련연구와의 비교 및 의의

지금까지 본 논문에서 제안한 TELOR는 역공학 방법론은 아니므로 역공학 방법론과 직접적으로 비교할 수 없으나 역공학을 이용한 모델링을 하는 연구는 유사하므로 기존의 몇몇 역공학 방법론과 비교 요약하면 표 5와 같다. 비교 대상이 되고 있는 역공학 방법론들은 앞서 2장 관련연구 부분에서 나왔던 표 1을 근거로 하였다. 또한 비교 대상이 되는 방법론의 종류로는 중복되는 방법론은 제외시키고 역공학 방법론 중에서 가장 특이할 만한 이론을 가진 것을 비교 대상으로 하였음을 밝혀둔다. 레거시 시스템의 의미 있는 정보가 객체 모형에

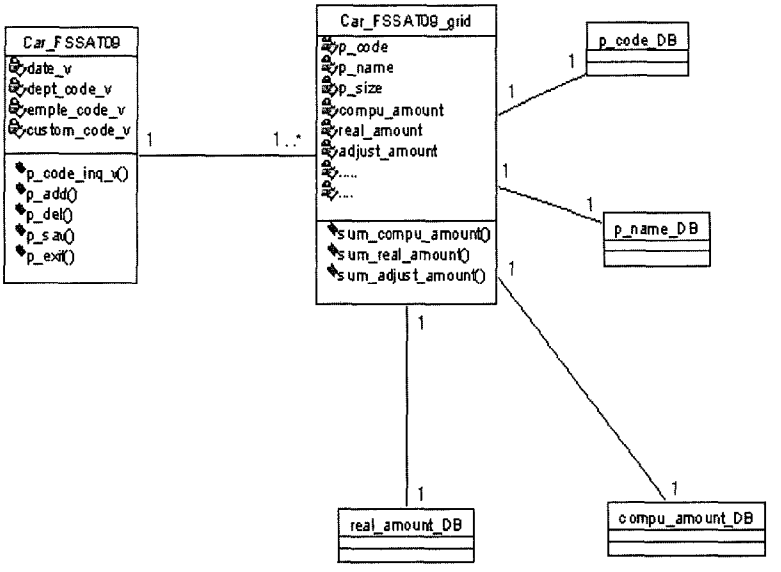

그림 18 차량재고 실사 인터페이스 구조

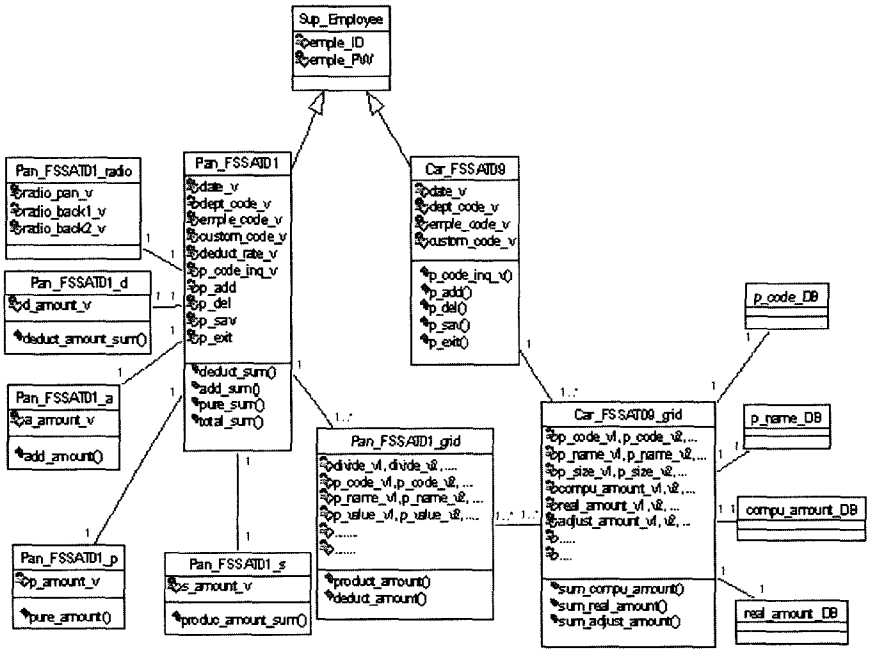

그림 19 통합된 객체 모델

의해 훨씬 잘 표현될 수 있는데 그 이유는 객체지향 기법이 다양한 표현을 할 수 있을 뿐만 아니라 현재까지는 가장 최신의 모델링 수단이기 때문이다. 그에 반해 기존 역공학 방법론들은 데이타 관점이나 또는 프로세스 관점 등의 어느 한 면만을 반영하기 때문에 객체지향 기법을 거의 반영하지 않고 객체지향 기법을 적용시켰다 하더라도 객체를 표현하는 모델링 언어가 표준을 따르지 않았던 것이 사실이다.

또한 기존 방법론들은 다양한 입력 데이타를 매개체로 이용하여 역공학을 시도하였다. 즉, DB 스키마(Chiang), 소스 프로그램(RE2), 실행 프로그램, 레거시 시스템의 모든 요소(RE2)들 등 다양하다. TELOR는 가장

표 5 기존 역공학 방법론과의 비교

방법론 / 특성	Chiang	RE2	FORE	Abdelwahab	TELOR
시스템 관점	데이타	프로세스	객체	객체	객체
목적	물리적 DB에서 의미 데이타 복구	레거시 시스템으로부터 재사용 모듈 생성	레거시 시스템으로부터 객체 구조 유도	소스 코드로부터 다른 툴에서도 상호동작 가능한 메타모델 생성	· 레거시 시스템으로부터 데이타, 프로세스를 이용한 객체구조 · 통합 모델 추출
입력 데이타	물리적 DB 스키마	레거시 시스템의 모든 자료	입력화면의 양식과 사용자의 상호작용 정보	객체지향 시스템의 소스코드 정보	· 레거시 시스템의 인터페이스 정보 · 레거시 시스템/사용자 간의 상호작용정보
목표 모델	EER 모델	특별한 모델 없음	ECRC를 이용한 객체 모형	UML에 기반한 CTF 모델	· UML에 기반한 객체(클래스) 다이어그램 · 통합 모델
가정 및 제한	· 3단계 정규화 · 필드 명을 기반으로 내포 종속성 생성 (synonyms/homonyms 파악)	대상 기준이 추상 데이타 타입을 구현한 재사용 모듈을 자동적으로 생성해주지 못해 사람 판여 요구	사용자 추론에 의해 객체의 내부 처리규칙을 파악하는 것으로 제한	소스코드 내용에 영향을 주지 않는 범위에서 trace 크기를 감소시켜야 함	사용자와 시스템의 상호작용 정보를 통한 내부처리 규칙을 파악하는 것으로 제한
인적참여 여부	필요	필요	유지보수 전문가에 의한 지원 필요	소스코드 분석 전문가 필요	분석전문가의 참여 필요
자동화 지원 여부	지원 안됨	Prolog 프로그램 사전	지원 안됨	지원 안됨	부분 지원

단순하면서도 사용자와 시스템간의 상호작용이 긴밀히 일어날 수 있는 매개체로 레거시 시스템의 인터페이스 정보를 선택하였다. 인터페이스 정보를 입력 매개체로 사용하는 것은 기존의 FORE나 Abdelwahab 방법론과 유사하나 결과적으로 추출되는 모델은 표준 모델링 언어가 아닌 CRC 표기법을 확장한 자체적인 표기법인 ECRC 표기법을 사용하거나 CTF라는 추상화된 메타모델을 생성하므로, 추출된 결과물을 현재의 시스템 체계에 이용하려면 객체지향 표기법으로 또다시 매핑(mapping)을 해야 하는 어려움이 있다. 이는 매핑하는 과정에서 발생할 수 있는 오류 및 정보 손실이 일어날 가능성이 크며, 추출모델이 불완전하다는 것을 말해준다. 이에 반해 TELOR는 UML 표기법에 따르는 객체(클래스) 다이어그램을 생성하고 이렇게 다양하게 추출된 객체정보들이 하나로 통합되는 통합 모델이 가능하다.

6. 결론 및 향후 연구

대부분의 기업에서는 비 객체지향적 또는 객체지향 언어를 사용하여 시스템을 개발하였으나 객체지향 개념이 정확히 적용되지 않은 레거시 애플리케이션 시스템을 그대로 사용하고 있다. 왜냐하면, 기업에서 사용하고 있는 현 시스템이 가장 안정적이기 때문이다. 그러나 이러한 기존의 시스템은 급변하는 업무 환경에 적합하게 대처할 수 없는 경우가 대부분이다

본 논문에서는 레거시 시스템의 인터페이스로부터 의미 있는 정보를 역공학적으로 객체를 분석해내고 그에 따른 통합 모델링을 추출해내기 위한 객체추출기법(TELOR)을 제안하였다. TELOR는 인터페이스의 구조 및 사용자와 시스템간의 상호작용 정보들을 활용하여 객체지향 모델을 생성한다. 즉, 레거시 시스템에 관한 어떠한 정보가 없더라도 사용자에 의해 시스템이 동작 가능하다면 TELOR에 의해 분석에 필요한 정보를 수집할 수 있으며 역공학을 이용하여 객체 단위로 정보를 추출해 낼 수 있다. 인터페이스에 관한 지식은 개발 중인 서브시스템을 통해 수집 가능한 정보로 가정하고 자료로 만들어 이용하였다. SFA 시스템에 적용하여 객체 모델을 추출해본 결과는 4장에서 나타내 보였다.

본 연구의 성과를 요약하면 다음과 같다. 첫째, 입력 자료는 사용자/시스템간의 가장 기본적으로 일어나는 인터페이스에 기반 한 상호작용 정보를 활용한 역공학 연구이고 둘째, 인터페이스에 관한 지식을 지식 저장소에

저장할 수 있는 방법을 알고리즘화 된 단계로 제시하였으며, 셋째, 데이타 및 프로세스를 동시에 다루고 있다는 것이고 넷째, 이러한 인터페이스 정보를 이용해 객체 분석 모델을 제시하였고 이를 SFA라는 실제 시스템에 적용하여 사례를 들었다. 따라서 결과적으로, 레거시 시스템을 유지보수 하거나 새로운 형태로 재공학 할 경우 본 TELOR가 완전하게 적용될 수 있을 것으로 기대된다.

향후 연구사항으로는 TELOR에서 보여준 객체 분석 과정을 자동화하여 시스템 분석 및 설계자의 업무를 좀 더 쉽게 처리할 수 있도록 자동화 도구의 개발이 필요하다.

참 고 문 헌

[1] P. Aiken, A. Muntz, and R. Richards, "A Framework for Reverse Engineering DoD Legacy Information Systems," Proceedings Working Conference on Reverse Engineering, IEEE Computer Society Press, pp. 180~191, 1993.

[2] Y-G. Kim, "Improving Legacy Systems Maintainability," Information Systems Management, pp. 7~11, 1997.

[3] K-H. Kim, Y-G. Kim, "Process Reverse Engineering for BPR: A Form-Based Approach," Information and Management, pp. 182~200, 1998.

[4] E. J. Chikofsky and J. H. Cross II, "Reverse Engineering and Design Recovery: a Taxonomy," IEEE Software, pp. 13~17, 1990.

[5] M. Malki, A. Flory, M. K. Rahmouni, "Static and Dynamic Reverse Engineering of Relational Database Applications: A Form-Driven Methodology," ACS/IEEE International Conference on Computer Systems and Applications, pp. 191~194, 2001.

[6] C. Batini, S. Ceri and S. B. Navathe, Conceptual Database Desing-An Entity Relationship Approach, Benjamin/Cummings, Redwood City, 1992.

[7] S. B. Navthe and A. M. Awong, "Abstracting Relational and Hierarchical Data with a Semantic Data Model," Proceedings 6th Int. Conference on the Entity-Relationship Approach, Springer-Verlag, pp. 305~336, 1987.

[8] R. H. L. Chiang, T. M. Barron, and V. C. Storey, "Reverse Engineering of Relational Database: Extraction of an EER Model from a Relational Database", Data Knowledge Engineering, 12(2):107~142, 1994.

[9] P. Shoval and N. Shreiber, "Database Reverse Engineering: from the Relational to the Binary Relationship Model," Data Knowledge Engineering, pp. 293~315, 1993.

[10] G. Canfora, A. Cimitile, and M. Munro, "A Reverse Engineering Method for Identifying Reusable Abstract Data Types," Proceedings Working Conference on Reverse Engineering, IEEE Computer Society Press, pp. 73~82, 1993.

[11] K. Saleh and A. Boujarwah, "Communications software reverse engineering: a semi-automatic approach," Information and Software Technology, pp. 379~390, 1996.

[12] H. Lee and Ch. Yoo, "A Form Driven Object-oriented Reverse Engineering Methodology," Information Systems, Vol. 25, pp. 235~259, May, 2000.

[13] Abdelwahab Hamou-Lhadj, Timothy C. Lethbridge "A Metamodel for Dynamic Information Generated from Object-Oriented Systems," Electronic Notes in Theoretical Computer Science, Volume 94, pp. 59~69, May, 2004.

[14] M. Kantola, H. Mannila, K-J. Raiha, and H. Siirtola, "Discovering Functional and Inclusion Dependencies in Relational Databases," International Journal of Intelligent Systems, pp. 591~607, 1992.

[15] N. Mfourga, "Extracting Entity-Relationship Schemas from Relational Databases: A Form-Driven Approach," Proceedings of the 4th Working Conference on Reverse Engineering, 6-8 Oct., pp. 184~193, 1997.

[16] H. Lee, C. Lee, and C. Yoo, "A Scenario-Based Object-Oriented Methodology for Developing Hypermedia Applications," 31st Proceedings of Hawaii International Conference on System Science, Vol. 2, IEEE Computer Society Press, pp. 47~56, 1998.

[17] R. H. L. Chiang, T. M. Barron, and V. C. Storey, "Performance Evaluation of Reverse Engineering Relational Database into Extended Entity-Relationship Models," Proceedings 12th Int. Conference on the Entity-Relationship Approach, Springer LNCS823, pp. 352~363, 1994.

[18] K. Saleh and A. Boujarwah, "Communications Software Reverse Engineering: a Semi-Automatic Approach," Information and Software Technology, Volume 38, Issue 6, pp. 379~390, 1996.

[19] L. H. Etzkorn, W. E. Hughes, Jr. and C. G. Davis, "Automated Reusability Quality Analysis of OO Legacy Software," Information and Software Technology, Vol. 43, Issue 5, pp. 295~308, 2001.

이 창 목

1999년 호원대학교 전자계산학과 졸업(이학사). 2001년 전북대학교 대학원 전산통계학과 졸업(이학석사). 2001년 3월~현재 전북대학교 대학원 컴퓨터통계정보학과 박사수료. 관심분야는 객체지향 소프트웨어 개발방법론, 소프트웨어 아키텍쳐, 소프트웨어 재사용 등

유 철 중
정보과학회논문지 : 소프트웨어 및 응용
제 31 권 제 1 호 참조

장 옥 배
정보과학회논문지 : 소프트웨어 및 응용
제 31 권 제 1 호 참조