역공학 기반 금융VAN 연동 서비스 모델링을 통한 시스템 구현 연구

김규원* 박보경 장우성 문소영** 김영철
홍익대학교 소프트웨어공학연구실*, (주)액트 원자력기술본부**
{kkw1206, bk, jang, bob}@selab.hongik.ac.kr

A Study on System Implementation through modeling the Financial VAN(Value Added Network) Connected Service Based on Reverse engineering

Kyu-Won Kim*, Bo-Kyung Park, Woo-Sung Jang, So-Young Moon** R. Young Chul Kim
Software Engineering Lab, Hongik University

요 약

VAN 시스템에서 운영되는 소프트웨어는 개발이 완료한 후에도 업체들의 다양한 요구 사항을 수용해야한다. 소프트웨어 개발 초기에 갖추어진 개발문서들은 오랜 유지보수 기간이 지나면 초기 개발의 내용과 많은 부분이 다르다. 이때 개발자가 각 각의 요구사항을 수용할 때 경험을 토대로 수정하기 때문에 개발문서가 정확하게 반영되지 않는다. 이러한 경우에 개발문서는 그 역할을 수행하지 못하므로 소스 코드에 의지 할 수밖에 없다. 또한 절차식 언어로 개발된 소프트웨어의 경우는 유지보수가 더 어렵다. 절차식 언어를 객체지향 언어로 변경한다면 유지보수성이 높아 질 것이다. 본 논문에서는 이러한 문제를 해결하기 위해 역공학을 이용하여 기존의 절차 식 언어를 객체 지향 언어로 변경하기 위한 과정을 제안한다.

1. 서 론

VAN사에서 근무하며 개발업무를 진행하는데 있어서 새로운 요구사항이 빈번하게 접수된다. 하지만, 그 시스템에 대한 이력과 정보가 정확히 문서화 되어있지 않으면, 금융 정보를 다루는 업무이기에 프로그램의 오동작에 대한 부담은 굉장히 크다. 이러한 경우 소스코드를 통해서만 시스템의 정보를 알 수 있기 때문에, 요구사항에 대한 것을 처리함에 있어서 기존에 운영되던 프로세스에 영향을 끼치지 않기 위해서 더 많은 시간을 할애하게 되고 소스코드의 복잡성이 증가 하게 된다.

절차식언어로 개발된 프로그램은 사용자의 요구사항 변경에 융통성 있는 대응이 쉽지 않고, 확장성이 적으며, 분석에서 설계로의 전환 시 정보가 손실되는 등의 문제점을 가지고 있다. 이전* 연구에서 C언어로 개발된 SYSTEM을 UML로 모델링 하여 객체 지향적인 SYSTEM으로 모델링 하는 연구를 진행 했었다. [1]
적용사례로서 채택한 VAN 연동 시스템은 실시간 신용카드 거래승인 시스템으로서, C 언어로 개발된 시스템이다. 항상 서비스 되어야 하는 시스템에 대한 정확한 문서가 없을 때에, 시스템의 업그레이드나 수정요청에 대한 대응은 쉽지 않은 작업이다. 이러한 문제점을 해결하기 위해서 시스템에 대한 가장 정확한 정보인 원시코드를, 역공학을 이용하여 시스템의 정보를 추출한다.[2] 이렇게 얻은 정보를 이용해 절차식 언어보다 유지보수성이 높은 객체지향 언어로의 전환을 시도 하였다.
객체 지향 시스템으로 변경하게 되면, UML로 모델링함으로서 시스템의 정보를 보다 이해하기 쉽게 문서화 할 수 있다. 본 논문에서는 C언어에서 다른 객체 지향 언어로의 Refactoring을 위하여 UML의 클래스 다이어그램을 이용하여 시스템을 추상화 하였다.[3][4] 이를 위한 모델링 도구로는 본 연구실에서 개발한 HiMEM을 사용하였다. HiMEM은 UML로 모델링을 한 다이어그램으로부터 코드까지 생성이 가능한 모델링 도구이다.[5]
본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구로서 Reverse Engineering과 Refactoring, 객체지향 프로그래밍, VAN에 대해 기술하고, 3장에서는 절차식 언어로부터 객체 지향 언어로 3단계에 걸쳐 변환하는 방법을, 4장에서는 적용사례로서 C언어로 개발된 VAN연동 SYSTEM의 객체 지향언어로의 변환, 그리고 마지막 장에서는 결론 및 향후 연구를 언급한다.

* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업(NIPA-2010-(C1090-0903-0004))과 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2010-0012117).

2. 관련 연구

2.1 Reverse Engineering 과 Refactoring

소프트웨어의 문서가 전혀 없어서 원시코드가 유일한 프로그램 이해의 단서가 되는 경우에 유지보수는 매우 어렵다. 원시코드로부터 다양한 정보를 만들어 내는 기술이 소프트웨어 역공학(Reverse Engineering)이다.
역공학은 설계도를 추출하거나 구현과는 독립적인 추상화된 표현을 만드는 것이다. 따라서 대상 시스템에 변경을 가하거나 새로운 시스템으로 변경이 아니라 분석인 것이다.[2]
Refactoring이란, 소프트웨어의 큰 변경 없이 내부적인 구조를 쉽게 이해하고, 경제적으로 수정할 수 있는 작업이다. 즉, 내부적인 요소들의 변경으로 시스템의 성능을 최적화시키는 작업이다. 기본적인 기능들의 변화 없이 성능을 향상시키고, 근본적인 원인 요소들만 찾아내서 변화시키기 때문에 작업이 용이하다. 시스템의 내부 구조의 개선이 이루어지지 않거나 외부 코드의 행위가 너무 오래 되었다면, 시스템의 구성 요소들에 교환을 통해 성능향상과 기능적인 행동을 보장하고, 근본적인 원인들만 변화시킴으로써 향상된 시스템을 만들어 준다.[3][4]

2.2 객체지향 프로그래밍

객체지향의 개념이 나오기 전 소프트웨어를 바라보는 시각은 하나의 프로그램은 필요한 데이터 구조와 이 데이터 구조 위에 수행되는 함수들의 집합이었다. 이것의 가장 큰 문제는 연관이 많은 데이터와 함수들도 별도의 독립된 것처럼 코드 상에 정의하는 것이다. 데이터 부분은 흔히 전역 변수들의 정의 부분에 포함되어 있고, 이 데이터 부분과 연관된 함수들은 다른 함수들과 함께 정의되어 연관 관계를 잘 표현 하지 못하였다. 반면에 객체 지향적 시각은 하나의 프로그램은 여러 개의 객체로 구성이 되며, 객체가 데이터 구조와 함수를 동시에 포함하고 있다는 것이다. [6]

2.3 VAN (Value Added Network)

VAN(Value Added Network)이란 전기 통신사업자로부터 회선을 차용하여 고도의 통신처리 기능 등 부가가치를 붙여서 제3자에게 재판매하는 통신망이라는 사전적인 의미가 있다. 국내에서 VAN이 의미하는 바는 신용카드에 관련된 통신업무 및 정보처리 관련 업무를 하는 사업자를 말한다. VAN서비스의 구성은 가맹점, VAN사, 카드사가 연계되어 이루어진다. VAN의 거래승인 흐름 구성도는 그림1과 같다.[8] 신용카드 단말기, 인터넷거래, 대형점의 HOST 서버 등은 전화선, 인터넷, 전용선 등 다양한 회선으로, 각각의 통신전문으로 VAN사의 승인 시스템으로 전송하면, VAN사는 각각의 요청전문에 대해서 발급한 카드사로 전송을 하여, 거래승인을 이루어지게 해준다.

그림1. VAN 승인 흐름도

3. Reverse Engineering 기법을 이용한 구조적 언어를 객체지향으로 변환방법

본 논문에서는 절차식 언어로 개발된 프로그램으로부터 UML 모델을 모델링하고, 객체지향 UML 모델을 모델링 변환하고, code를 생성하는 과정을 적용하였다. UML에 사용되는 Diagram 중 Class Diagram을 이용한다. Class Diagram은 시스템을 추상화하기 용이하며, 시스템의 스켈레톤 코드를 제공해주는 diagram이다.
그림2는 기존에 개발되어 있는 C언어로 구성된 시스템을 모델링과 모델변환, 객체 지향적 클래스 다이어그램으로 변환으로 새로운 시스템을 구현하는 방법을 나타낸 그림이다.

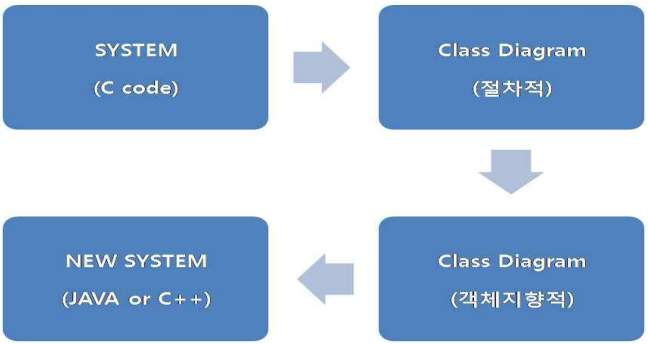

그림2. 시스템의 변환 과정

3.1 C언어로부터의 Class Diagram 생성

C언어에는 클래스가 존재하지 않는다. 따라서 C언어를 class diagram으로 표현하기 위해서는 기준을 정해야 한다. C언어의 구성요소는 변수, 구조체, 함수로 이루어지게 된다. Class는 클래스이름, 속성, 메소드로 이루어진다. C언어는 함수 호출로서 동작하는 프로그래밍 언어이고, 객체지향 언어는 객체의 상태변화로 인해 프로그램이 동작 하게 된다.
먼저, C언어의 함수의 호출로서 동작하는 프로그래밍 언어이기 때문에, class 생성기준을 함수로서 정의한다.

첫째, 함수의 이름을 클래스의 명칭으로 정의한다.
둘째, 사용된 parameter, 변수를 속성 값으로 정의한다.
셋째, 함수간의 호출 관계로서 클래스의 관계를 상속 관계를

정의하는데, 다른 함수를 호출하지 않는 함수는 자신을 호출하는 함수의 메소드로서 정의한다.

넷째, 구조체는 멤버들의 값을 설정하는 메소드를 하나 정의하고, 그 구조체를 사용하는 함수와 has-a 관계로서 그 자체를 하나의 클래스로 정의 한다.

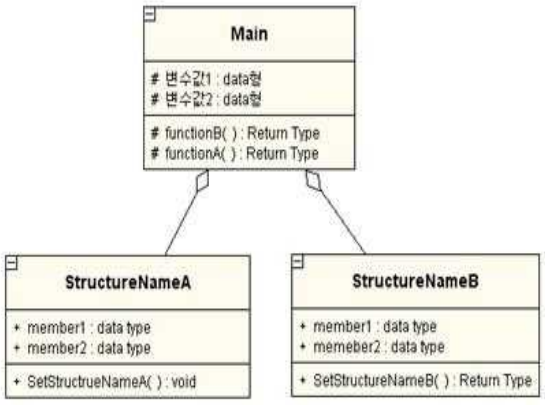

그림3. 절차지향 class diagram

그림3은 변수값 두 개와, structure 2개를 가지고 있고, 2개의 함수를 호출하는 C언어 프로그램을 class 다이어 그램으로 나타낸 것이다.

3.2 Model to Model

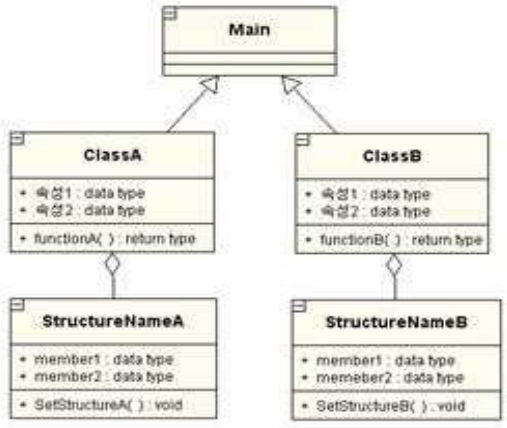

그림4. 객체지향 class diagram

그림4는 그림3의 model을 객체 지향적으로 변환 한 모습이다. Main 클래스에서 모든 것을 포함하던 구조에서 각각의 기능을 하는 function들에 대해서 각각의 ClassA, ClassB로 정의 하고, function 에서 사용되는 구조체 A, B에 대해서 정의한 Class A, Class B와의 관계쉽을 정의 하였다.

하나의 class가 많은 속성 값들과 메소드를 모두 포함하는 것은 객체 지향 관점에서 올바르지 않는 것이다. 관계가 있는 속성과 method를 묶어 여러 클래스로 분리해서 다이어그램을 구성하였다. 이러한 모델변환을 통한 클래스 다이어그램은 C코드에서 객체지향 언어로의 Refactoring을 용이하게 해준다.

3.3 Model to Code

Model 변형을 통해서 얻은 class diagram은 class name, 속성, 메소드들이 직접 코드로 매핑 될 수 있다. 맵핑된 코드는 메소드의 내용은 정의 되어있지 않은 스켈레톤 코드이다.

그림5는 class diagram으로 부터 생성한 소스 코드의 모습이다. C언어로 구현되어 있는 코드에서 객체지향 언어로의 변환은 모델링과 모델 변환을 통하여 보다 쉽게 Refactoring이 가능하게 된다.

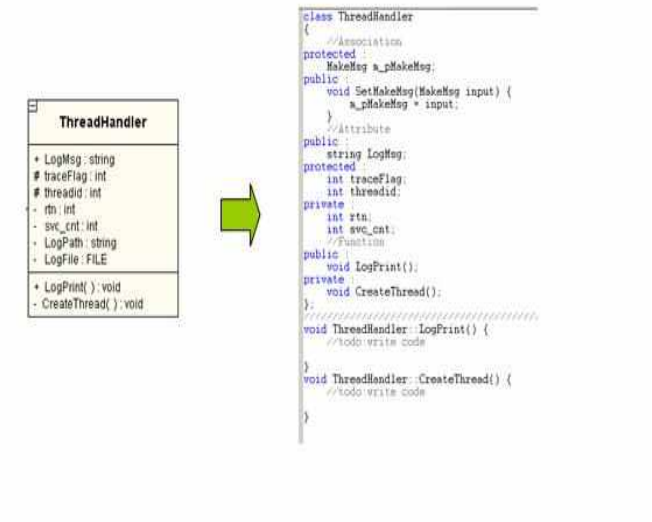

그림5. 코드 변환

4. 적용 사례

VAN 연동 시스템에서는 다음과 같은 요구 사항을 가지게 된다.

첫째, 하나의 거래는 반드시 Unique 한 키 값을 가진다.

둘째, 짧은 시간에 발생하는 다수의 거래에 대하여 정확하고 빠른 처리가 가능해야 한다.

셋째, 예외적인 구조를 가지는 요청에 대한 처리

넷째, 응답을 받지 못했을 경우의 Timeout 처리

다섯째, 거래내역은 데이터베이스에 저장되어, 정산업무가 가능 하도록 설계가 되어야 한다.

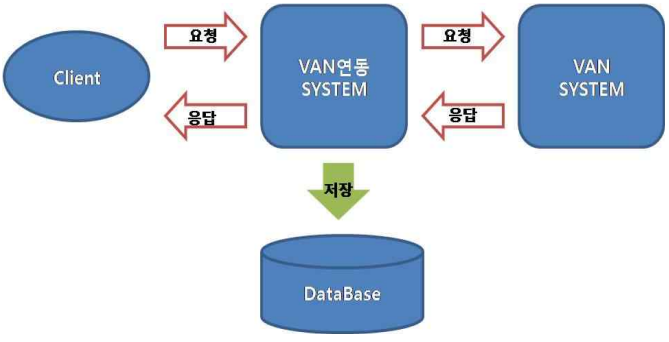

그림6. VAN 연동 시스템

C언어로 개발된 VAN(value added network) 연동 시스템을 객체 지향 언어인 C++로 변환 해본다.
그림6은 client로부터 데이터를 수신하여 VAN과의 중간에서의 데이터의 흐름을 나타낸다. Client는 VAN 연동 SYSTEM으로 거래승인을 요청하게 되면, VAN연동 SYSTEM은 VAN사와의 정의 되어있는 구조의 통신전문으로 승인 요청전문을 전송한다.
VAN사에서는 수신된 전문을 카드사로부터의 승인여부를 VAN연동 SYSTEM에 응답을 주고, VAN사로부터 응답전문을 수신한 VAN연동 SYSTEM은 거래승인 데이터를 DB에 저장하고 Client로 응답전문을 전송한다.
그림7은 C언어로 개발된 VAN 연동 시스템을 class diagram으로 표현 한 것이다. 이와 같은 모델링은 절차식 언어를 class diagram 으로서, 객체지향적인 관점으로 보았을 때, 바람직하지 않은 class diagram의 모습으로 나타내어진다.

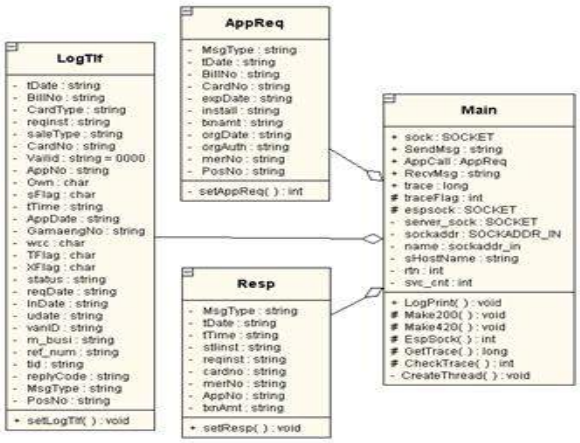

그림7. 절차식언어의 class diagram

그림8은 그림7에서 modeling한 class diagram으로부터 객체 지향 적으로 modeling이 변화되는 모습을 표현한 것이다. subMain()안에서 거의 모든 기능을 하던 구조에서, 각 기능별로 class를 나누어 class들의 관계를 class diagram으로 modeling 하였다.

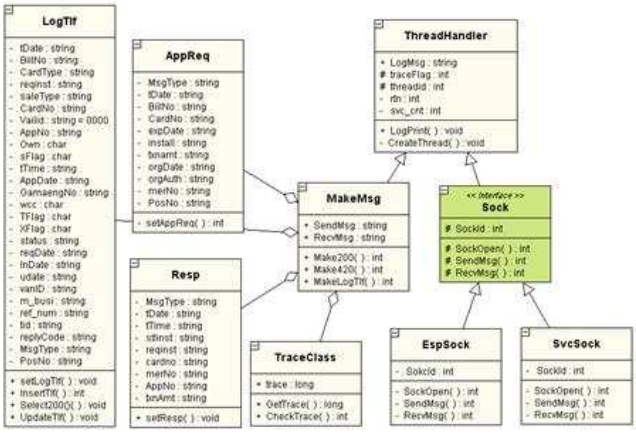

그림8. 객체지향 class diagram

이렇게 생성된 객체지향 프로그램은 요구사항 변경에 융통성이 좋고, 프로그램 분석에 용이하며, 추후에 효율적인 설계를 하기가 용이하다.
HiMEM 도구는 class diagram으로부터, JAVA, C++, C 코드 등을 추출 할 수 있는 기능이 있고, 그림9는 코드로 사용 할 언어를 선택하는 것이다.

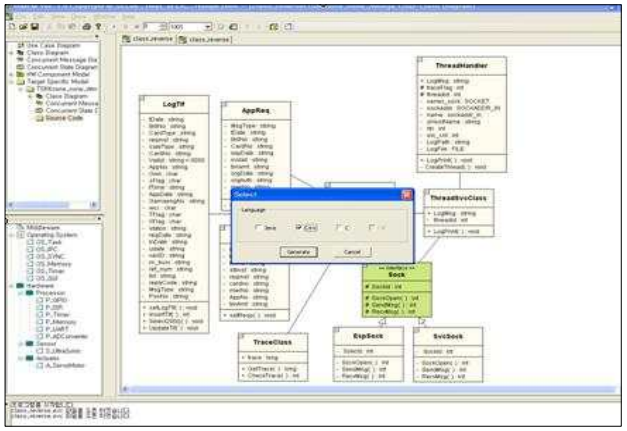

그림9. HiMEM의 코드 추출모습

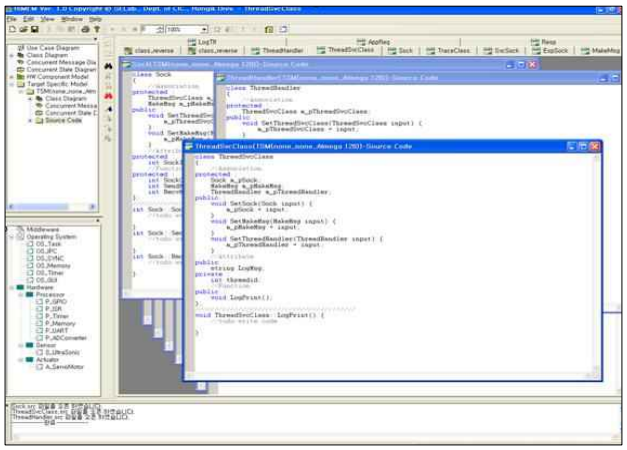

그림10. HiMEM으로 추출된 코드

그림10은 C++를 선택해서 얻은 소스 코드의 추출 모습이다.

5. 결론 및 향후 연구과제

본 논문에서는 절차식 언어로 개발되어 있는 시스템을 객체지향 시스템으로 변경하기 위해, UML의 클래스 다이어그램을 이용하였다.
C언어로 개발되어 있는 프로그램을 Reverse Engineering을 통해 정보를 추출하기 위해서 UML의 클래스 다이어그램을 이용하였다.
절차지향 클래스 다이어그램을 먼저 모델링하고, 모델 변환을 통하여 객체 지향적 클래스 다이어그램으로 변경한 후 객체 지향 언어로 Refactoring하는 과정을 적용하였다. 이러한 과정들에서 모델링 도구로는 본 연구실에서 개발한 HiMEM 도구를 사용하였다. HiMEM은 모델링한 diagram으로부터 코드를 발생시킬 수 있는 도구

이다.

하지만, Class Diagram만으로는 시스템의 스켈레톤 코드만을 얻을 수 있기 때문에, 향후의 연구에서는 state diagram, sequence diagram 등의 모델링을 통하여 실제 구동 가능한 source 코드를 추출하여, 기존의 시스템에 비해서 시스템의 성능, 요구사항에 대한 유연성, 확장성 등을 비교해볼 예정이다.

참고문헌

[1] 김규원,김영철, “금융VAN (Value Added Network) 연동 서비스를 위한 모델링 연구” 한국인터넷 방송 TV 학회 : 추계 학술 대회, 2009. 12

[2] 최은만 저, “객체지향 소프트웨어 공학“

[3] 이종호,박진호,류성렬, “객체지향 기반의 Refactoring 프로세스“, 정보과학회 논문지 컴퓨팅의 실제7권 4호, 2001.8

[4] 박진호,이종호,류성렬,“소프트웨어 유지보수와 재사용을 위한 재공학 Refactoring 기법”. 한국정보과학회 2000 년도 봄 학술발표논문집 제27권 제1호(A), 2000. 4

[5] 김우열, 김동호, 문소영, 김영철,” xUML을 사용한 MDA 기반 임베디드 소프트웨어 컴포넌트 시스템을 위한 설계 재사용”, 한국정보과학회 2005 가을 학술 발표 문집(Ⅱ)제32 제2호, 2005. 11

[6] 이준경,조창현,이동길,최완,송영기,김영시, “객체지향 프로그래밍 개념 및 패러다임”, [ETRI]전자통신 동향 분석-제8권 제1호,1993.3

[7] Bernd BURUEGGE, Allen H.DUTOIT,“Object-Oriented Software Engineering, Using UML, Patterns, and java”

[8] 황삼생, 양재영, “VAN을 기반으로 한 컴퓨터와 네트워크에 의한 전자 상거래 연구”. 한국국제경영관리학회 2001년도 학술발표대회 논문집, 2001. 11