통합 객체 관리 모델을 위한 F77/J++ 생성기에 관한 연구

선 수 균† · 송 영 재††

요 약

최근 컴퓨팅 환경은 통합되는 개방형 시스템으로 변모하고 있다. 요즘에도 여러 기업과 연구기관에서는 노후코드를 그대로 사용하고 있으며 급변하는 컴퓨팅 환경에 능동적으로 대처를 못하고 있다. 또한 여러 개발자가 소프트웨어 개발에서 생산성 문제와 노후코드를 변환시키는데 많은 고민을 하고 있다. 이에 본 논문에서는 급변하는 환경에 대처하고 새로운 소프트웨어 개발에 생산성을 향상시키기 위해서 통합 객체 관리 모델을 제안한다. 이 모델은 세 계층으로 나누며 위 계층은 사용자에게 정보를 분류하고 디스플레이하는 계층이다. 가운데 계층은 제어기능으로 통합 관리기 계층이다. 아래 계층은 데이터를 관리하는 객체 관리 저장소계층이다. 따라서, 통합 객체 관리 모델을 위한 F77/J++ 생성기(FORTRAN-77/Java Code generator)를 설계 구현한다. 생성기는 노후코드를 새로운 코드로 변환시켜 생산성을 향상시키고 업무를 재 설계하는데 도움을 준다. 제공학 전략을 적용하여 아홉 단계 전략으로 구성되어 있다. 이는 향후 시스템의 통합에 잇점인 소프트웨어의 재사용성을 극대화하여 생산성을 향상시키는 프로토타이핑을 지원할 것으로 기대된다.

A Study on F77/J++ Code Generator for Integration Object Management Model

Su-Kyun Sun† · Yong-Jea Song††

ABSTRACT

Lately computing environment is changing into integrating open system. Some corporations and research institutions are still using old codes and not dealing with the rapid changing environment actively. Also several software developers have difficulties with the problems of productivity and translating old codes.

This paper proposes Integration Object Management Model to deal with the rapid changing environment effectively and to improve productivity about new software development. The model is divided into three layers. the first layer classifies and displays information to users, the second layer controls function, the integration and management layer, and the last layer manages data, the object management storage layer. So it designs and implements F77/J++ Generator system(FORTRAN-77/Java code generator) for Integrated Object Management Model. The generator helps to translate old codes into new codes in redesigning the business and promoting productivity. It consists of nine-stage strategies using reengineering. This might support afterward prototyping in maximizing the reuse of software, which is advantage to the integration of the system, and in promoting its productivity.

1. 서 론

객체 지향 패러다임의 확산으로 인하여 소프트웨어 개발을 위한 객체 모델의 사용이 일반화되고 있으며 아울러 객체 지향 패러다임을 지원하는 자동화 도구의 사용이 확산되고 있다[1].

최근에는 여러 기업과 연구기관은 포트란 코드를 50 퍼센트 그대로 사용하고 있으며 이러한 단체들은 급변

† 종신회원 : 동원대학 사무자동화과 교수
†† 종신회원 : 경희대학교 전자계산공학과 교수
논문접수 : 2000년 6월 10일, 심사완료 : 2000년 10월 6일

하는 컴퓨팅 환경에 능동적으로 대처를 못하고 있다. 또한 여러 개발자가 소프트웨어 개발에서 생산성 문제와 노후코드를 변환시키는데 많은 어려움에 처에 있다[1,2]. 소프트웨어를 개발하는 과정도 발전하여 개념정립, 설계, 코딩, 시험의 과정을 밟아 하나의 기능을 구현하는 개발 사이클이 응용서비스의 요구속도를 만족할 수가 없이지고 소프트웨어 개발 속도의 단축 및 다양한 기능요구를 만족할 수가 없다. 그래서 소프트웨어의 중요성이 21세기에도 더욱 강조되고 있고 소프트웨어 생명주기에도 상당한 관심을 가지고 있다. 또한 웹 환경하에서 소프트웨어 개발을 통합하는 방법과 기존의 노후코드를 새로운 소프트웨어로 자동 생성하는 연구가 집중되고 있다. 이로서 새로운 소프트웨어를 짧은 시간에 분석하고 특징을 파악하여 생산성을 높이려는 노력이 가중되고 있다[3, 15].

최근에 제시된 방법인 Microsoft의 Repository는 Microsoft의 관계 DBMS인 Access와 SQL Server를 이용하여 산출물에 대한 동시 공유와 효율적인 재사용을 지원하고 있다[13]. 그러나, 이 제품은 Microsoft 플랫폼에 종속적이며, 현재 Visual Basic 개발 환경을 위한 컴포넌트(component)의 지원과 탐색을 위주로 지원하고 있어 C++이나 Java와 같은 다양한 개발 환경을 지원하지 못하고 있고, 다양한 산출물을 모두 관리하지 못하는 단점이 있다. 이러한 여러 단점을 극복할 수 가 있는 모델이 절실히 필요하게 되었다.

따라서 본 논문에서는 급변하는 소프트웨어 환경에 대처하고 객체관리를 효율적으로 관리하는 새로운 모델 즉, 소프트웨어 개발 속도의 단축 및 다양한 기능요구를 만족할 수 있는 모델인 통합 객체 관리 모델을 제안한다.

이 모델은 세 계층으로 나누이져 있으며 첫 번째 계층은 사용자에게 정보를 분류하고 디스플레이하는 계층, 두 번째 계층은 제이기능으로 통합 관리기계층이며, 세 번째 계층은 데이터를 관리하여 객체 지향 소프트웨어 공학 프로세스에 의해 생성되는 제반 산출물을 객체 형태로 통합 관리하는 객체 관리 저장소(object management repository) 계층이다.

본 논문에서는 이 세 번째 계층에 속하는 생성기를 설계, 구현하려고 한다. 이것이 웹 환경에 맞는 F77/J++ 생성기(FORTRAN-77/Java Code generator)이다.

본 논문의 목적은 다음과 같다. 첫 번째는 통합 관리함으로서 차세대 소프트웨이 개발할 때 개발 속도의 단축 및 다양한 기능요구를 만족하여 생산성 향상시키기 위한 통합 모델제시이다 두 번째는 급변하는 환경에 대처하고 객체관리를 효율적으로 유지, 관리하여 개발에 필요한 산출물들을 저장할 수 있는 모델제시이다. 세 번째는 인터넷 사용인구의 폭발로 다양한 서비스 요구에 대한 소프트웨이 개발 필요성은 대단히 높은데 비해 고전적인 소프트웨어 개발 속도로는 더 이상 해결방법이 없으므로 이것을 해결할 수 있는 모델제시이다. 또한, 이 모델을 중심으로 한 생성기를 제시하여, 기존의 노후코드를 새로운 코드로 생성하는 단계를 재공학 전략으로 소프트웨어 개발자에게 노후 코드에서 새로운 코드로 변환하는 과정을 제시한다

본 논문에서 제안한 모델의 장점은 모델을 통하여 다양한 CASE 도구와 사용자들이 여러 산출물들을 동시에 공유하게 함으로써, 객체 지향 소프트웨이 공학 프로세스에서 발생하는 다양한 산출물을 데이터베이스화하여 필요한 산출물을 용이하게 검색하는 객체 관리 저장소를 제시한 것이다. 또한 제어기능으로 기존 모델과 이기종간 모델을 결합할 수 있는 객체 관리기 제시이다.

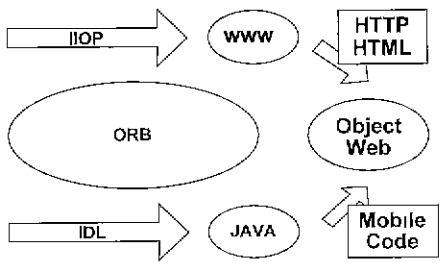
2. 관련 연구

J Ewer는 소프트웨어 공학 연구(CASE study)를 집중적으로 연구하고 있으며 노후코드를 변환하는 단계로 구분하여 설계 구현하였고 연구 평가한 결과도 상당한 효과가 있었다[7]. 여기에서는 재공학 기법을 적용했으며 포트란 CFD(Computational Fluid Dynamic)코드로 언급했으며, 비객체지향 코드를 객체지향 코드로 변형하는 과정을 구현까지 상세하게 설명하고 있다. 포트란으로 만들어진 시스템을 C++ 코드로 구성된 시스템으로 변환하는 것은 구조적 패러다임을 객체지향 패러다임으로 변환하는 것을 의미하는데 [7]에서는 단계별로 설명하고, 재사용 가치를 시뮬레이션을 등해서 통계적으로 보여주고 있다. 또한 여기에서 중요하게 인급되는 것은 C++로 변환하는 시스템을 CASE Tool로 구현하였다

최근 연구는 프로그램 변환을 지원하기 위해 자동화된 소스코드로 재구성(re-structuring)하는 툴로 발전하고 있으며 이러한 툴이 Sage++이다[3]. Sage++는 포트란코드를 재공학 기법으로 생성할 수 있는 소프트웨어 툴로 'pretty-print'라는 소스코드를 사용되었다 이

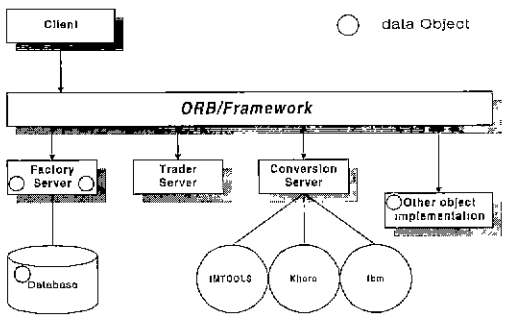
것은 필요한 정보를 생성하는데 13단계의 설계를 거치면서 사용되는데 많은 시간이 소요되는 단점이 있다. 이러한 정보는 서브루틴에서 요구되는 변수와 루틴을 자동으로 생성하지만 제어기능이 없다. 또한 효과적으로 관리해 주는 객체를 재사용할 수 있는 객체 저장소가 없어서 효과적인 관리를 할 수가 없다.

Angus와 Curtis의 논문에서는 수치적인(Numerical code) 코드 포트란을 C++ 코드로 재공학 기법으로 구현하였다. 여기서는 재공학의 여러 단계의 장점과 단점을 자세하게 묘사하고 있다[1] 통합 환경에 적합한 구조가 CORBA와 ObjectWeb이다. ObjectWeb이란 웹과 CORBA, 그리고 자바에서 제공하는 다양한 분산 서비스들과 객체지향 패러다임을 이용하기 위해, 시스템 변형 없이, 웹에 CORBA를 통합한 환경을 말한다. (그림 1)은 ObjectWeb의 구성을 나타낸 것이다[6, 12].



(그림 1) ObectWeb의 구성

DISCUS(Data Interchange and Synergistic Collateral Usage Study)는 미국정부가 추진한 최초의 CORBA관련 프로젝트로서 분산 객체들 간의 상호 운용성을 보장하고, 시스템 통합에 필요한 설계 및 구현 비용을 절약하기 위한 프레임워크를 개발하는 데 있으며, 개발된 프레임워크는 (그림 2)와 같다[8].



(그림 2) DISCUS의 구조

3. 통합 객체 관리 모델

최근 컴퓨팅 환경은 통합되는 개방형 시스템으로 변모하고 있다. [4, 15]연구에서는 서비스 품질 향상을 위한 제어 기능 부재로 효과적인 관리를 할 수가 없었다. 또한 재사용 할 수 있도록 보관, 유지 관리하는 객체 저장소가 없고 이기종간 통합할 수 있는 구조가 불가능하여 웹과 연동 할 수 있는 환경이 매우 미약했다.

따라서 본 논문에서는 세 계층으로 구성된 통합 객체 관리 모델(Integrated object Management Model)을 제시 하고자 한다. 이 모델은 [15]에서 제안한 모델을 더 확장시킨 것이다

첫 번째는 프로세싱 관리 계층이다 산출물의 분류, 설계, 분석, 적용에 필요한 곳을 다룬다.

두 번째는 통합 관리기 계층이다. 기존 시스템과 통합 및 표준화 작업, 산출물들을 제어하는 제어기능으로 분산 개체 환경에 적합한 구조로 이기종간 쉽게 통합할 수 있다. 통합 관리기(I-CASE M)를 두어 이기종간 시스템을 통합 관리하며 클라이언트와 서버의 프로세서와 데이터를 통합 관리하고 통합에 필요한 Object Web와 CORBA[6, 11, 12]에 기반을 둔 시스템 통합 관리 모델이다

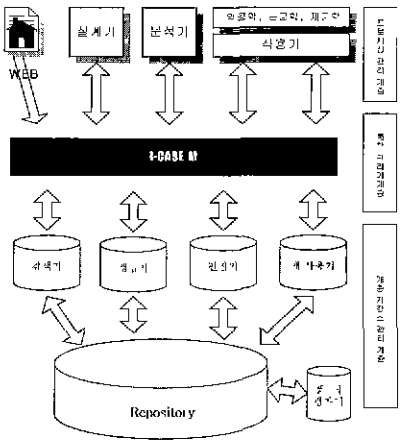
세 번째는 객체 저장소 관리 계층이다. 이것은 급변하는 소프트웨어 환경에 대처하고 객체관리를 효율적으로 관리하며 개발에 필요한 산출물들을 저장할 수 있는 객체저장소 관리 계층이다 즉, 데이터를 관리하고 객체 지향 소프트웨어 공학 프로세스에 의해 생성되는 제반 산출물을 객체 형태로 통합 관리하는 데이터베이스이다.

통합 객체 관리 모델은 소프트웨어 생명주기의 효율적인 관리를 하며, 분산객체 환경에 적합한 구조를 만드는데 중점을 두고 소프트웨어 개발에서 생성된 산출물들을 효율적으로 관리하기 위한 새로운 모델이다

(그림 3)은 통합 객체 관리 모델 전체 구성을 나타낸 것으로 세 가지 계층으로 구성되는데, 위 계층을 프로세싱 관리 계층, 가운데 계층을 통합 관리기 계층(I CASE Manager), 아래 계층을 객체 저장소 관리 계층이다.

위 계층은 프로세싱 관리 계층은 (그림 3)에서 웹, 설계기, 분석기, 적용기를 포함하고 있다. 사용자에게 정보를 분류하고, 디스플레이하는 계층이다. 가운데 계층은 제어기능으로 (그림 3)에서 "I-CASE M"부분을

확대해 나타낸 것이 (그림 4)으로 통합 관리기 계층이다. 아래 계층은 객체 저장소 관리 계층으로 (그림 3)에서 검색기, 생성기, 편집기, 재사용기, 저장소(Repository)부분으로 데이터를 관리하며 객체 지향 소프트웨어 공학 프로세스에 의해 생성되는 제반 산출물을 객체 형태로 통합 관리하는 객체 저장소 관리(object repository management)계층으로 관계형 데이터베이스화 한 구조이다



(그림 3) 통합 객체 관리 모델 전체 구성도

본 논문에서 제안한 모델의 장점은 첫 째는 소프트웨어 생명주기 전체를 이해하는데 용이하다. 둘째는 기존 모델과 이기종간 쉽게 결합할 수 있는 모델제시다. 세 째는 소프트웨어 개발자에게 재사용을 극대화 할 수 있는 모델 제시다. 네 번째는 웹 환경에 쉽게 데이터를 분류할 수 있도록 하는 환경 제시다. 다섯 번째는 통합 객체 저장소 제시다. 여섯 번째는 향후 소프트웨어 컴포넌트 중심의 모델과 쉽게 통합 할 수 있는 모델 제시다 본 논문에서는 F77/J++ 생성기(FORTRAN-77/Java Code generator)를 설계 구현한다. 이것은 아래 계층에 속하는 생성기를 설계, 구현한 것으로 노후코드(Legacy code)를 여러 단계를 거쳐 웹 환경에 맞는 새로운 코드(New code)로 변환하여 단순한 코드변환에만 그치는 것이 아니고, 새로운 환경에 적응 할 수 있는 분석단계에서 코드의 특징을 손쉽게 알 수 있고 활용해서 설계단계까지 적용할 수 있다.

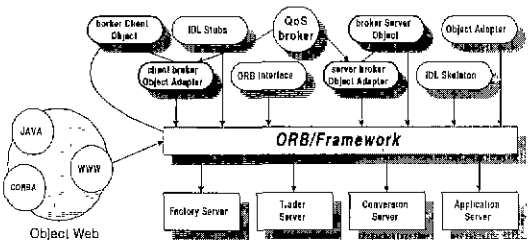
3.1 프로세싱 관리 계층(Processing Management layer)

소프트웨어 생명주기 관리를 하기 위한 생명주기 서비스를 제공하고 객체지향 소프트웨어 개발과정 중에는 문서, 다이아그램, 원시코드, 방법론 정보, 사용자 인터페이스 객체, 데이터 베이스 스키마 객체 등의 다양한 산출물이 있는데 이런 산출물들을 분류하는 방법을 제시하는 것이 프로세싱 관리 계층이라 하고 분류 객체라 한다. 분류를 View 객체라 하는데 사용자에게 정보를 출력하는 객체이다. 즉 여러 산출물들을 어디에 저장시킬 것인가를 결정시키고, 재사용할 수 있도록 분류하고 저장하는 단계를 결정하는 것이 프로세싱 관리 계층이다

3.2 통합 관리기 계층(I-CASE M layer)

분산 객체 시스템에서 클라이언트와 서버에 각각 프로세스와 데이터의 형과 불균형으로 통합에 필요한 적절한 아키텍처가 필요하게 되었다. 응용들간의 시스템 통합[15]을 위한 프레임워크가 부족하여 분산환경에서 기존의 소프트웨어를 재사용 할 수 없는 단점이 있었다 통합 관리기는 이러한 단점을 보완하고 프레임워크 기반의 통합은 실제 통합에 필요한 설계 구현 비용을 최소화 할 수 있다. 통합 관리기(I-CASE Manage)의 구조를 나타낸 것이 (그림 4)과 같다

통합 관리기 구조로 이기종간 시스템을 통합시킬 수 있으며, 제어기능을 추가함으로서, 소프트웨어 환경을 통합하여 차세대 소프트웨어개발을 활성화시키고 개발기간을 단축시켜 소프트웨어 산출물들을 효율적으로 유지보수, 저장, 관리하여 생산성을 극대화시키는 것이다.



(그림 4) 통합 관리기(I-CASE M) 구성도

통합 객체 관리기(I-CASE M)는 ObjectWeb이란 소프트웨어 구조를 사용하고 웹와 CORBA로 인결하여 통합 관리하는데 필요한 구조를 만들어 통합 관리를 용이하게 했다. CORBA는 분산 컴퓨터 환경과 이 기종 분산 환경의 시스템통합을 위한 표준이다.

3.3 객체 저장소 관리 계층(Object repository Management layer)

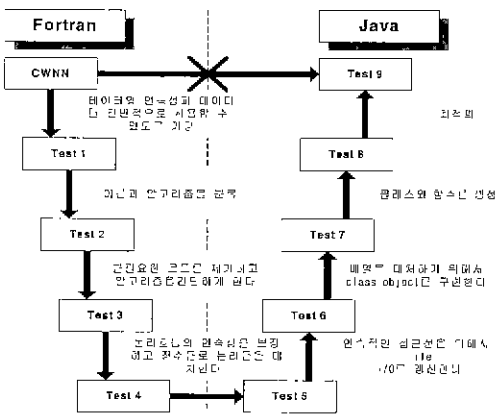
객체 지향 소프트웨어 공학 프로세스에서 발생하는 다양한 산출물을 데이터베이스화하여 필요한 산출물을 용이하게 검색하기 위한 것이 객체 저장소 관리 계층이다. 이 객체 저장소 관리 계층은 다양한 도구와 사용자의 동시 공유를 위한 병행 처리, 주요 산출물 객체에 대한 등급별 접근 처리와 보완처리, 시스템 장애 발생에 대한 회복, 객체의 올바른 상태를 유지하기 위한 무결성 처리 기능이 제공되어야 한다. 이러한 기능은 객체 저장소를 파일 시스템을 기반으로 하지 않고, DBMS를 기반으로 개발함으로써 제공받을 수 있다.

4. F77/J++ 생성기 설계

본 논문에서는 통합 객체 관리 모델을 중심으로 생성기를 설계 구현한다. 기존의 노후코드를 웹 환경에 적합한 새로운 코드로 자동 생성하여 시스템을 통합하고, 재사용하여 소프트웨어의 생산성 향상을 추구하는 것이다. 생성기는 노후코드를 객체지향 환경으로 전환하기 위한 전 단계로 새로운 코드로 변환시키는 것이다. <표 1>는 재공학 측면에서 설계 단계를 표로 나타낸 것이다. 본 논문에서는 제안된 모델을 중심으로 포트란(FORTRAN77 : F77)언어를 자바(Java : J++)언어로 변환하는 F77/J++ 생성기를 설계, 구현하려고 한다. 기존의 많은 코드를 다양한 개발 환경에 맞는 새로운 코드로 생성하는 최적의 생성기를 만드는 것도 객체지향 소프트웨어 공학 프로세서에 의해 생성되는 산출물이다. 이러한 산출물들을 통합 관리하고, 유지보수하기 위해서는 통합 관리하는 관리기가 필요하게 되었고, 이런 산출물들을 필요할 때마다 검색하고, 재사용하기 위해서는 데이터베이스화하여 필요한 산출물을 효율적으로 검색하는 것이 객체 저장소이다. F77/J++ 자동 생성기의 전체 시스템 구조는 다음 (그림 6)과 같으며 아홉 단계로 구성되고 있다. 재공학(re-engineering)으로 적용하여 노후코드를 재구성하여 새로운 언어로 제작성 한다면 소프트웨어 재사용성을 극대화하고 그로 인한 생산성을 높일 수 있을 것으로 기대 된다. 따라서 본 논문에서는 노후코드인 F77에서 웹 환경에서 구현할 수 있는 자바(Java)언어로 부분 자동 생성하는 생성기를 설계함으로써 재구성하는 단계이며 멀티 단계 프로세서로 설계하는 과정을 다루기로 한다 (그림 6)에서 수직라인(Central vertical line)은 포트란과 J++ 사이에 경계표시를 나타낸 것이다. F77/J++ 생성기 설계단계는 아홉가지로 이루어져 있고 체공학적인 전략으로 구성되어 있으며 또한 각 단계마다 변형과정으로 나눌 수 있다.

<표 1> 9가지 처리기

	처 리 형 태
1	데이터 연속성과 데이터를 가공
2	이름과 알고리즘을 분류
3	불 필요한 코드 제거
4	대처하기
5	변 환
6	파일 입,출력을 갱신
7	클래스 객체론 구현
8	클래스와 함수를 생성
9	최 적 화



(그림 5) 단계별 프로세스 구성

첫 번째 단계는 데이터 연속성(data consistency)을 확실하게 하고 데이터를 전반적으로 사용 할 수 있도록 포괄적인 데이터를 만든다.

두 번째 단계는 이름과 알고리즘을 분류한다.

세 번째 단계는 과다한 코드를 제거하는 것과 단순화(simplification)시키는 것이다. 즉 불필요한 코드를 제거하고 알고리즘을 간단하게 한다.

네 번째 단계는 논리흐름의 연속성을 보장하고 정수들로 논리들을 대처한다. 모든 논리적인(logical) 변수를 변환시키고 제어의 견실한 사용이 확실하게 한다. 실행할 수 있는 문장보다 오히려 CONTINUE 문장을 사용한다. 이것은 J++로 변환하는데 도움이 되고 다른 loop 루틴을 만드는데 쉽게 해 준다. 예로써 "IF(<expr>) <statement>"가 "IF (<expr>) THEN <statement> ENDIF"으

로 대신함으로써 다음의 J++로 변형이 쉽게 만들어질 것이다.

다섯 번째 단계는 포트란을 J++ 프로시저로 변환하는 단계를 말한다. 이 단계는 네 번째 단계가 완전하게 이루어질 때 포트란 소스코드는 동시적으로 실행되면서 J++ 프로시저로 변환되고 각 항목은 변환에 맞는 루틴으로 변환할 필요가 있다.

여섯 번째 단계는 연속적인 접근성을 위해서 파일 입출력을 갱신한다. 호환성(compatibility)을 위하여 모든 입력 출력 파일을 수정하고 재구성한다. 포트란과 J++에서 입출력의 헤더가 다르기 때문에 입력부분에서 마지막부분을 초기화하는 FileInputstream.getline().함수를 사용하는 것이 필요하다. 포트란과 J++ 언어가 서로 다르기 때문에 새로운 코드(new code)를 어떤 지역(area)이든 발전시켜야 한다.

일곱 번째 단계는 배열을 대처하기 위해서 클래스 객체(object)를 구현한다. 프로시저 루틴(procedural routines)를 위하여 클래스 메소드(class method)를 생성한다.

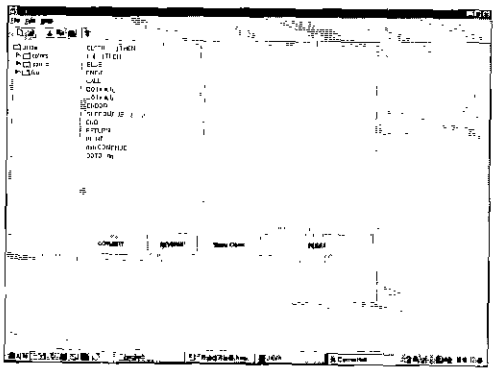
여덟 번째 단계는 프로시저 루틴(procedural routines)을 위한 클래스 넘버 함수(class member functions) 생성이다 여기에서는 클래스와 함수를 새로운 코드에 맞게 생성한다.

아홉 번째 단계는 마지막 단계로 최적화를 이루는 것이다 최초로 class 객체 J++버전은 통계적으로 객체들의 선언된 배열로 사용된다.

5. F77/J++ 생성기 구현

[15]에서 연구과제였던 자동 코드 생성 기능으로 노후코드(Legacy code)를 새로운 코드로 생성하여 소프트웨어 개발자가 새로운 개념을 쉽게 습득한다. 본 논문에서는 (그림 7)에서 F77/J++ 생성기의 변환하는 초기 화면으로 가운데 화면은 포트란 언어를 입력한 생태를 나타낸 것으로 사용자 인터페이스를 자바로 구현하여 웹 환경에서 쉽게 적용할 수 있다.

본 논문에서는 포트란 코드를 자바로 자동 변형하는 것으로 노후코드를 재구성하는 방법을 나타내는 것이다. 여기서 멀티 단계에서 네 번째 단계는 포트란코드가 상세 설계로 변형이 되고 호환성(compatibility)이 있다. F77/J++ 생성기 구현단계는 아홉 가지의 재공학 전략으로 이루어지며 변형과정으로 나눌 수 있다.



(그림 6) F77/J++생성기 의 변환기

첫 번째 단계는 데이터 일관성을 확실하게 하고, 포괄적인 데이터를 만든다 동질성이 없는 원시 파일은 다른 파일로 결합된다 열린 파일(Open file)의 숫자를 허용하는데 제한이 있기 때문에 이 단계가 필요하다. 생성기에서는 이 단계가 필요하며 철저하게 사용된다. 객체지향 형태에서 소프트웨어를 재구조하기 위해서는 데이터는 클래스별로 모여 있어야 하고 추출할 수 있어야 한다. 파일을 포함한 비표준 포트란은 루틴사이에 pass data를 사용한다 이것들은 파일을 포함하는데 포함된 COMMON 데이터 선언이다 단지 각 변수를 위해서 존재된 하나의 선언이다

(리스트 1)는 COMMON 블록 안에 데이터 파싱을 하는 예제를 나타낸 것이다.

```
Legacy FORTRAN code
CALL BUOYAN (RMETHD, ELEMAT    ELEVOL, TEMPER, ..)
SUBROUTIME BUOYAN( RMETHD,     ELEMAT, VOLUME, T, .. )
INTEGER ELEMAT (TOTELE)
REAL  VOLUME(TOTELE), T(TOTELE)
B = T(I) + ..
becomes
INCLUDE 'DATABASE. INC'
CALL BUOYAN ( RMETHD, )
SUBROUTINE BUOYAN( RMETHD, . )
INCLUDE 'DATABASE.INC'
B = TEMPER(I) * ...

with DATABASE. INC defined as
INTER  ELEMAT(TOTELE)
REAL ELEVOL(TOTELE), TEMPER(TOTELE)
COMMON / CELL-D/ ELEMAT, ELEVOL,  TEMPER
```

(리스트 1) COMMON과 파일로 된 Passing data

데이터 항목들은 관련된 항목을 이름 공통 블록(name common blocks)으로 포함한다 서브루틴을 포

함한 더미 아큐먼트(argument) 이름은 새롭게 정의된 COMMON 배열 변수들로 직접 대신 할 수 있다 (리스트 2)는 코드 조각(code fragment)으로 지적된 것을 어떻게 배열 인덱스 값이 배열 요소들로 대신하는가를 나타내고 있다

```
Legacy FORTARAN
CALL HBOUND ( H ( ICELL) , ..)
SUBROUTINE HBOUND (HVAL, .. )
REAL HVAL
HVAL = HVAL * ...

becomes
CALL HBOUND ( ICELL, . )
SUBROUTINE HBOUND ( ICELL, .. )
INCLUDE 'DATABASE INC'
INTEGER ICELL
H (ICELL) = H(ICELL) ...
```

(리스트 2) 아큐먼트 파싱하는 것

두 번째 단계로는 이름과 알고리즘을 설명한다.

기능적인 의미(functional meaning)와 사용을 전달한다. (리스트 3)는 최초의 이름을 변환하는 기능을 나타낸 것이다

Legacy FORTRAN	*New naming convention*
MCSOLV	solve_momentum
CALGEN	calc_generation_rate
RDINFF	read_inform_file
H	read_inform_file
TEMPER	enthalpy
U	temperature
KINET	Cu_velocity kinetic_energy
DISSIP	dissipation_rate

(리스트 3) 코오드 분류를 위한 이름 변환

세 번째 단계는 과다한 코드를 제거하는 것과 단순화(simplification)시키는 것이다.

Legacy code fragment	*Equivalent code abstracted*
MITERS = MAXITR(4)	NTEGER VAR_W_VELOCITY
CALL SORSCH(.)	PARAMETER(VAR_W_VELOCITY = 4)
SERROR(4) = RESIDU	CALL SORSCH(VAR_W_VELOCITY, ..)
RELAXA = VRELAX(4)	CALL LINRLX(VAR_W_VELOCITY,)
CALL LINRLX(. .)	
VARERR(4) = RESIDU	

(리스트 4) 재배치와 추상화를 위한 할당

네 번째 단계는 모든 논리(logical) 변수를 변환시키고 제어의 견실한 사용이 확실하게 한다

실행할 수 있는 문장보다 오히려 CONTINUE 문장을 사용한다. 이것은 J++로 변환하는데 도움이 되고 다른 loop 루틴을 만드는데 쉽게 해 준다. 예로써 "IF (<expr>) <statement>"가 "IF(<expr>) THEN <statement> ENDIF"으로 대신함으로써 다음의 J++로 변형이 쉽게 만들어질 것이다. 다음으로는 "IF (<expr>) GO TO <label>"가 이것이 종종 do… while 구성으로 되기 때문에 왼쪽에는 변형이 안된다. "DO <label> <block> <label>CONTINUE"처럼 표준 loop 구조를 사용하는 구조는 비 표준구조인 "DO <black> END DO"로 과대한 사용을 회피하는 구조로 변환된다.

다섯 번째 단계는 포트란을 J++ 프로시저로 변환하는 단계를 말한다 이 단계에서는 네 번째 단계가 완전하게 이루어질 때 포트란 소스코드는 동시적으로 실행되면서 J++ 프로시저로 변환되고 각 항목은 변환에 맞는 루틴으로 변환할 필요가 있다. 여기에서는 lex[9]와 yacc[10]을 참조하여 파싱(parsing)과 컴파일러(compiler)로 이루어진다

egacy FORTRAN	*Macro replacement*
ELSEIF (. .) THEN	} else if ( ) {
IF ( ) THEN	if (.) {
ELSE	} else {
ENDIF	}
CALL	/* CALL REMOVED */
DO I = a, b, c	for (I=a, I>=min(a,b) && I<=max(a,b); I+=C) {
DO I = a, b	for (I=a, I<=b, I++) {
ENDDO	}
SUBROUTINE . (.)	void . (.) {
END	}
RETURN	return;
PRINT* , . .	System.out.println(.);
nnn CONTINUE	Label_nnn:
GOTO nnn	goto Label nnn;

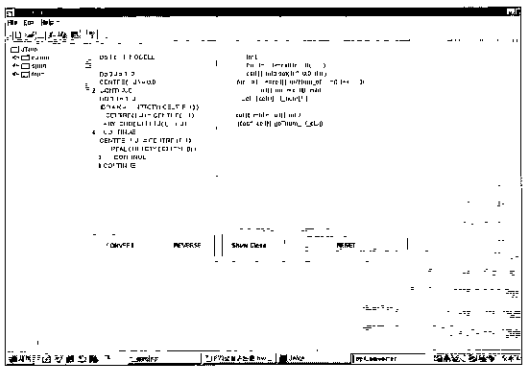
(리스트 5) 매크로 대처

여섯 번째 단계는 호환성(compatibility)을 위하여 모든 입출력 파일을 수정하고 재구성한다. 모든 입출력은 정확하게 수행된다 포트란과 J++에서 입출력의 해더가 다르기 때문에 입력부분에서 마지막부분을 초기화하는 FileInputstream.getline().함수를 사용하는 것이 필요하다. 포트란과 J++ 언어가 서로 다르기 때문에 새로운 코드(new code)를 만들기 위해서는 변수를 사

용하고 변수를 지정함에 있어 어떤 지역(area)이든 활용할 수 있어야 한다

```
Original legacy code                    New Java code using a vector class
   Do 1 I = 1, NOCELL                      int i ;
   Do 2 J = 1, 3                           for ( i=1 , i<=cell_length, i++ )
   CENTR(I 1, J ) = 0.0                   cell [i] mid.set( 0.0, 0.0, 0.0 )
2  CONTINUE                                for ( j=1   j<=cell[i] getNum_of_pts(),  ++
     )(
     DO 3 J = 1, 3                                cell[i] mid = cell[i] mid -
     DO 4 K - 1, NPTCTY( CELTYP( I ) )                    point[ cell[i] pt_num[j]  ];
          CENTRE( I, J ) = CENTRE( I, J )              }
+     - XYZCRD( CELPTS( I, K ), J )                cell[i] .mid = cell[i] mid /
4     CONTINUE                                    (float)  cell[i] .getTnum_of_pts();
     CENTRE ( I, J ) - CENTRE ( I, J )
+        / REAL ( NPTCTY ( CELTYP (I) ) )
   3         CONTINUE
   1 CONTINUE
```

(리스트 6) vector algebra를 위한 클래스 사용하는 예



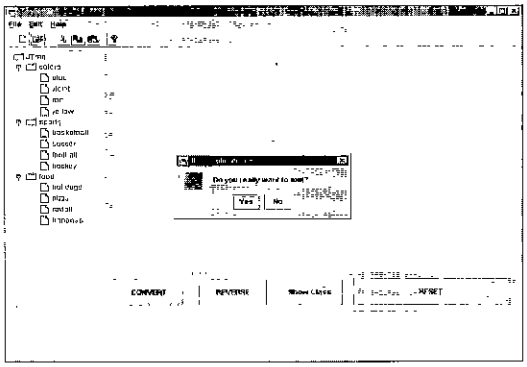
(그림 7) F77/J++생성기 의 실행화면

(그림 8)은 F77/J++ 생성기의 실행화면을 나타낸 것으로 Vector algebra를 위한 클래스를 사용한 것이다.

일곱 번째 단계는 프로시저 루틴(procedural routines)을 위하여 클래스 메소드(class method)를 생성한다.

여덟 번째 및 아홉 번째 단계는 최적화와 개선으로 이루어진다. 처음에는 각 객체(Object)마다 클래스(class)로 구성하며 각 객체는 동적으로 구성된 배열로 구성되고 있고 최적화로 마지막 단계로 부족한 부분을 수정한다 에러 출력을 사용하는 것도 여기 단계에서 사용된다 동적으로 선언된 배열은 쉽게 구현되고 수정도 쉽게 할 수 있다. 또한 각 객체가 요구에 생성되이지고 소멸되어지는 것을 위해서 객체에서 포인터의 배열이 구현하기가 가능하다.

(그림 9)는 F77/J++ 생성기의 종료화면을 나타낸 것이다



(그림 8) 변환기 종료화면

6. 적용 사례

제안한 모델을 중심으로 F77/J++ 생성기 적용 사례를 제시하고자 한다. 기존의 시스템인 자동차 관리는 포트란 언어로 구성돼 있다. 노후코드에 사용된 코드의 구성은 107 소스화일인데 158루틴을 포함하고 있다. 총 라인 수는 22,450 라인으로 구성되이 있다. 따라서 본 논문에서는 많은 코드를 표시할 수 없어서 생략하여 보여준다. 새로운 코드로 변형하여 AMOSS(Auto-Mobile Customer Service Shop)로 구현한 과정을 단계별로 시간을 체크했으며 실험 결과와 분석자료를 살펴보면 다음과 같다 <표 2>는 하드웨어를 비교한 것이다

〈표 2〉 하드웨어 비교

Hardware,	Compiler and	System
586DX PC,	Salford FORTRAN v2.74,	Legacy
586DX PC,	Java 1.2,	New
Sun classic,	f77 v3.0, -O3-cg89,	Legacy
Sun classic,	J++ 1.2,	New

새로운 시스템(New system)으로 변형하는 과정을 단계별로 구조화되지 않은 시스템에 많은 시간을 연구 분석에 소요했다 각 단계를 소요시간으로 계산한 결과를 <표 3>로 나타낸 것이다 종합적으로 볼 때 F77/J++ 생성기를 이용하면 변환과정이 85%가 자동화로 이루어진다.

노후 시스템에서 새로운 시스템으로 변형하는 과정을 서로 비교하여 시뮬레이션 결과를 나타낸 것이다. 분석 통계로 보면 구조화되지 않은 시스템을 연구 분

석과정에서 많은 시간이 소요됨을 본다. 이것은 개발 과정에서 분석 단계가 얼마나 중요한 것을 보여준 것이다. 또한 본 논문에서 구현한 생성기는 각 단계를 거쳐서 마지막부분으로 프로그래머 작업으로 수정하는 단계는 자동화 단계가 끝나고 최종적으로 컴파일 했을 때 변수와 크기지정, 함수값 미지정으로 에러가 발생 시 프로그래머가 수정하는 것을 말한다

〈표 3〉 각 단계의 소요시간 분석

단계	처리 형태	소요 시간
	구조화되지 않은 시스템 연구 분석	30
	툴을 사용하기 위한 계획과 배우기	15
1	데이터 연속성과 데이터를 가공	15
2	이름과 알고리즘을 분류	10
3	불 필요한 코드 제거	15
4	대치하기	10
5	변 환	10
6	파일 입.출력를 갱신	10
7	클래스 객체를 구현	15
8	클래스와 함수를 생성	20
9	최 적 화	20
	프로그래머 작업	30

<표 4>는 노후 시스템과 새로운 시스템에서 586DX PC에서 일정한 루틴을 각 시스템에 적용하여 100번 반복하였을 때 소요되는 시간을 나타낸 것이다

〈표 4〉 586DX PC에서 100번 반복 소요시간

System	Compiler	Time
586DX PC,	Salford FORTRAN v2.74,	12m07s
586DX PC,	Java 1.2,	20m16s

자동차 관리 응용 프로그램인 고객관리, 사원관리는 포트란 언어를 사용돼 있는 것을 자바언어로 변형해서 웹 환경에 적용시킬 수 있게 변형시켰으며, 새로운 환경에 쉽게 적용할 수 있는 동기 부여와 소프트웨어 개발 시간을 단축시켰다는 장점이 있다.

새로운 시스템은 네 개의 소스와 13 헤더화일과 395 루틴을 포함하고 있는 클래스 함수로 구성되어 있다. 소스 화일 안에는 11,250 라인이 있고 1400 헤더화일이 있다.

<표 5>는 노후 시스템과 새로운 시스템에서 Sun시스템에서 일정한 루틴을 각 시스템에 적용하여 100번 반복하였을 때 소요되는 시간을 나타낸 것이다.

구조적 사항은 RDBMS를 사용하여 고객정보를 관리하고 JDBC Thin Driver를 사용하여 DB와 연동한다. 또한 연결된 DB를 JDBC를 이용하여 DBMS를 원격지에서 효과적으로 사용한다. 이 새로운 시스템은 GUI에서 DB로 제공이 요구되는 데이터는 새로운 고객과 자동차 정보와 고객관리의 갱신, 삭제, 삽입이 가능하다. JFC로 Swing을 사용하기 위한 패키지 정의하였으며 JDBC thin driver를 사용하였다

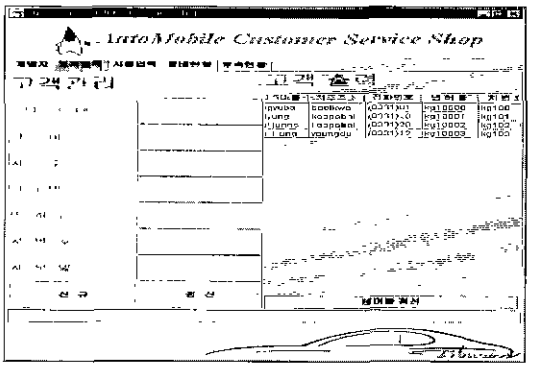
〈표 5〉 Sun 시스템에서 100번 반복 소요시간

System	Compiler	Time
Sun classic,	f77 v3.0, -O3-cg89.	5m10s
Sun classic,	J++ 1.2,	7m17s

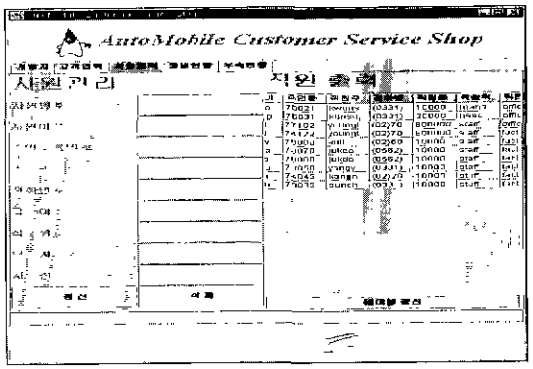
```
import com.sun.java.swing.*;        import java.awt.*;
import java.awt.event.*;
public class CustomerPanel_1 extends JPanel implements ActionListener
{ public Connection com;
  public String data[];              public JTextField text[];
  public JLabel field[];             public boolean check_bu[];
  public Statement myStatement;      public ResultSet results;
  public final int elem = 7;    public Hashtable hash;
  public CarPanel car;          public CustomerPanel_1(Connection con)
                {                    com = con;
  hash = new Hashtable();     hash.put("0", "social_security_num");
  hash.put("1", "owner_name");    hash.put("2", "address");
                ...
```

(리스트 7) 새로운 코드로 구성된 AMOSS

F77/J++ 생성기를 적용해서 포트란 언어로 작성한 응용 프로그램인 고객관리(그림 10), (그림 11)를 웹 환경에 알맞은 언어로 변형해서 구현한 것이다



(그림 9) 고객관리 화면



(그림 10) 사원관리 화면

7. 결 론

본 논문에서는 통합 객체 관리 모델을 제시한다. 기존 시스템은 소프트웨어 생명주기의 효율적인 관리와 분산객체 환경에서 소프트웨어 산출물들과 연결, 제어 메카니즘 부재로 인해 객체 저장소에 효율적인 저장이 어렵고 전체를 파악할 수 가 없는 단점이 있었다. 이런 단점들을 개선하기 위해서 ObjectWeb와 CORBA에 기반을 두고 전체를 파악할 수 있는 통합 객체 관리 모델을 제시한 것이다

제안한 통합 객체 관리모델을 적용하여 F77/J++ 생성기(Code generator)를 설계하고 기존의 응용 프로그램을 적용하여 시스템을 변환하는 과정을 구현한다. 이 시스템의 장점은 웹 환경에 쉽게 적용시킬 수 있으며, 자동코드 생성 기능으로 시스템 소프트웨어 개발 시간을 단축시킬 수 있고, 생산성을 높일 수 있다 또한 새로운 코드의 특징을 쉽게 파악하여 효율적인 유지보수관리, 소프트웨어 모듈의 재사용성을 증대시킬 수 있다 그러나 포트란77을 Java로 변환시 킬 프로그램들이 현 시점에서 얼마나 있는지가 단점이고, 좀 더 빠른 시간에 F77/J++ 생성기가 나왔더라면 실용성 측면에 있어서 아쉬움이 있다. 향후 연구과제로는 프로세서 계층에서 분류하는 알고리즘을 제시해야하고, 객체 지향 소프트웨어 공학 프로세스에서 발생하는 다양한 산출물을 데이터베이스화하여 필요한 산출물을 용이하게 검색할 수 있는 추출 알고리즘과 객체 관리 저장소에 관한 연구이고, 최근에 가장 많이 사용되는 다른언어(C,C++)를 Java언어로 변환하는 생성기 연구가 이루어져야할 것이다.

참 고 문 헌

[1] Angus, I. & Curtis, W., "From Fortran to object orientation experience with a production flutter analysis code," OONSKI'94 Conf. Proc., Oregon, USA., pp.174-80.

[2] Boehm, B., "Software engineering," IEEE Trans Comput., Chap.25. No.12, pp.1226-41. 1976.

[3] Bodin, F. et al, Sage++ : "An object-oriented toolkit and class library for building Fortran and C++ restructuring tools," OONSKI '94 Conf.Proc., USA., pp.122-36.

[4] Bernstein, P. A., and Dayal,U., "An Overview of Repository Technology," in Proc. VLDB, pp.705-713, 1994.

[5] Bernstein. P. A., "The Microsoft Repository," in Proc. VLDB, pp.3-12, 1997.

[6] B. Morgan, "CORBA meets Java," http://www.javaworld. om/javaworld/jw-10-1997 jw-10-orbajava.tml

[7] J Ewer, B. Knight & D. Cowell, Case study : an incremental approach to re-engineering a legacy FORTRAN Computational Fluid Dynamic code in C++, May 1995.

[8] I Jacobson, "Object-Oriented Software Engineering," Addison-Wesley Publishing Company, Inc., 1992.

[9] Kernigan, B. & Wilson, B., lex - a lexical analysis tool, The C Programming Language, Prentice-Hall, 1988.

[10] Kernigan, B. & Wilson, B , yacc - yet another compiler, The C Programming Language, Prentice-Hall, 1988.

[11] OMG, "CORBA2.0 Specification," http://www.omg.org/corba/iiop.html, 1995.

[12] Orifali, R. and Harkey, D., Client/Server Programming with JAVA and CORBA, Jon Wiley & Sons, 1997.

[13] Rumbaugh, J et, Object-Oriented Modeling and Design, Prentice-Hall, 1991

[14] 선수균, 송영재, "Java기반 ObjectWeb을 이용한 통합 시스템 설계", 98년 추계 학술발표집 한국정보과학회, 1998.

[15] 선수균, 송영재, "통합객체지향 관리기 중점을 둔 F77/J++ 생성기 설계", 99년 추계 학술발표집 한국정보과학회, 1999.

선 수 균

e-mail : sksun@Tongwon.ac.kr

1988년 경희대학교 공과대학 전자계산공학과(학사)

1988년~1996년 경희대학교 전자계산소 근무

1994년 경희대학교 대학원 전자계산공학과(공학석사)

1996년~1997년 영월전문대학 전자계산과 전임강사 및 전자계산소 부소장

1997년~1999년 동원대학사무지동화과 전임강사

1999년~현재 동원대학 사무자동화과 조교수

1998년 경희대학교 대학원 전자계산공학과 박사 수료

관심분야 : 소프트웨어 공학, S/W 재사용,CASE 도구, 통합 객체지향 시스템

송 영 재

e-mail : yjsong@nms.kyunghee.ac.kr

1969년 인하대학교 전기공학과 (공학사)

1976년 일본 Keio University 전산학과(공학석사)

1979년 명지대학교 대학원 졸업 (공학박사)

1971년~1973년 일본 Toyo Seiko 연구원

1982년~1983년 미국 Univ. of Maryland 전산학과 연구교수

1985년~1989년 IEEE Computer Society 한국지회부회장

1984년~1989년 경희대학교 전자계산소장

1976년~현재 경희대학교 전자계산공학과 교수

1993년~1995년 경희대학교 교무처장

1996년~1998년 경희대학교 공과대학장

1998년~현재 경희대학교 기획조정실장

관심분야 : 소프트웨어공학, OOP/S, CASE 도구, S/W 개발도구론, S/W 재사용