분산 객체 프레임웍을 지원하기 위한 재공학 시스템의 설계

조성림 [0] 이은주 이기열 우치수

서울대학교 컴퓨터공학부

{srcho, banny, kylee, wuchisu}@selab.snu.ac.kr

A Reengineering System Design for Supporting Distributed Object Framework

Sungrim Cho [0] Eunjoo Lee Keeyoull Lee Chisu Wu

School of Computer Science and Engineering

Seoul National University

요 약

소프트웨어 응용분야에서 재사용을 통한 프레임웍 기반의 개발 기술이 발전하고 있으며, 특히 CORBA와 같은 분산 객체 환경과 GUI를 지원하는 프레임웍들의 구현 및 응용기술에 대한 연구가 이루어지고 있다. 본 논문에서는 객체 지향 언어인 C++로 만들어진 독립적인 시스템 또는 클라이언트/서버 환경의 시스템을 CORBA를 이용한 분산 객체 프레임웍 환경으로 이주시키는 재공학 시스템 설계를 제안한다. 이 시스템을 이용하여 레거시 시스템을 재공학 하면 신뢰성과 상호 운용성이 높은 프레임웍 기반의 소프트웨어 개발을 효과적으로 지원할 수 있다

1. 서론

소프트웨어를 여러 응용분야에서 재사용이 가능하도록 컴포넌트 라이브러리나 응용 프레임웍의 형태로 개발하는 기술이 개발되고 있으며, 특히 CORBA와 인터넷 웹 기반 분산 객체 환경과 GUI를 지원하는 프레임웍들의 구현 및 응용기술이 빠르게 발전하고 있다.

기존의 유용한 레거시(Legacy) 시스템과 재사용성이 높은 프레임웍들을 상호 운용이 가능하도록 응용 도메인 별로 구축 또는 재구성하여 이들을 다수의 응용 소프트웨어의 개발에 활용할 수 있으면 매우 효율적인 소프트웨어 개발이 가능하다. 이에 따라 기존의 레거시 시스템을 이러한 프레임웍 환경에 맞게 재공학 하는 기술에 대한 필요성도 날로 높아지고 있다. 이러한 컴포넌트/프레임웍 기반의 소프트웨어 개발방식에는 응용 소프트웨어의 개발을 위해서 재사용되는 컴포넌트/프레임웍들과 재공학된 레거시 소프트웨어들에게 매우 높은 신뢰성, 견고성, 상호 운영성을 요구한다.

본 논문에서는 객체 지향 언어인 C++로 만들어진 독립적인 시스템 또는 클라이언트/서버 환경의 시스템을 CORBA를 이용한 분산 객체 프레임웍 환경으로 이주시키는 재공학 시스템 설계를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 레거시 시스템을 분산 객체 프레임웍 환경으로 재공학하는 데 관련된 기존 연구를 추상형태의 정의, 역공학, CORBA환경으로의 재구성 방법 등 3가지로 나눠서 살펴본다. 3장에서는 레거시 시스템에서 분산 객체 프레임웍로 이주시키는 전체 시스템 개요를 정의하고, 4장에서 역공학 서브시스템을, 5장에서 재공학 서브시스템을 설명한다. 마지막 6장에서는 결론과 함께 향후 연구과제를 기술 한다.

본 연구는 한국과학재단 목적기초연구(R01-1999-00238)지원으로 수행되었음.

2. 관련 연구

본 장에서는 레거시 시스템을 분산 객체 프레임웍 시스템으로 이주(migration)하기 위한 기존의 관련 연구들을 추상 형태의 정의, 역공학(*Reverse Engineering*), CORBA 환경으로의 재구성 방법의 3가지로 나눠서 살펴본다.

2.1. 추상 형태의 정의

기존의 레거시 시스템을 추상적인 형태로 표현하는 연구가 많이 수행되었다. 그러나, 대부분의 연구들은 소스 코드로부터 정적인 정보를 찾아내는 데 그치고 있다. 실제 CORBA 환경으로 재공학하기 위해서는 동적인 정보를 찾아내는 방법에 대한 연구가 필요하다.

Chen등은 reachability분석과 dead code를 찾아내기 위한 C++ 데이터 모델을 제시하였다[1]. 이 모델은 C++ 시스템에서 타입, 함수, 변수, 매크로, 파일 등 5가지 개체를 찾아내고, 상속, 프렌드쉽(*Friendship*), 포함(Containment), 템플릿 실현(Template instantiation), 참조등의 관계를 추상구문트리(Abstract Syntax Tree)로 나타낸다.

Jackson과 Rollins는 역공학을 위해 프로그램 의존관계를 표현하는 모델을 제시하였다[2]. 기존의 프로그램 의존관계 그래프의 문제점은 의존 관계를 표현할 때 불필요한 자료를 포함하고 있기 때문에 분석할 때 많은 시간과 노력을 필요로 하였다. 그러나, 이들은 불필요한 자료를 제거하여 프로그램 의존관계를 나타내는 그래프를 간략하게 표현하여 효과적으로 분석할 수 있는 모델을 만들었다.

2.2. 역공학

기존 역공학에 관한 연구는 프로그램 이해 지원 도구를 위한 연구들이 대부분이어서, 소스 코드로부터 정보를 추출하여 사용자에게 보기 좋은 형태로 보여주는 것을 목표로 한다. 그러나, 본 논문에서 제안한 시스템에서는 CORBA환경에 맞도록 바꾸어야 하므로, 정형적인(formal) 표현 방식으로 바꾸는 것을 목표로 한다.

Koskimies와 Mossenbock는 객체 지향 언어인 Oberon으로 이루어진 레거시 프로그램에서 시나리오 다이어그램을 추출하는 기법을 설명하고 있다[3]. 이 방법에서는 소스코드에 Instrumented Code(in new, call return)를 삽입하여 컴파일 후 실행시키고, Instrumented Code를 추적하여 정보를 추출하고 시나리오 다이어그램을 완성한다.

Kung등은 객체 지향 프로그램의 테스트를 위한 역공학 방법을 기술하였다[4]. 이 방법은 클래스와 멤버 함수들의 테스트 순서를 정하고, 멤버 함수들의 테스트 케이스를 만들고, 객체 상태에 의존하는 행위들과 행위들 사이의 관계에 대한 테스트 케이스등을 만드는 데 이용할 수 있다.

2.3. CORBA 환경으로의 재구성 방법

Parodi는 레거시 시스템, 상용 패키지 또는 하부 구조 등을 새로운 하이브리드 비즈니스 응용 프로그램에 맞도록 만들어 주는 방법을 제시했다[5]. 기존 시스템 자료의 80%이상이 메인 프레임에 있으므로, 이 자료들을 랩퍼(Wrapper)로 둘러쌈으로써 인터페이스 엔진을 통해 시스템의 재사용이 가능하도록 한다. 이 방법은 기존 시스템에 많은 수정을 가하지 않고도 새로운 CORBA 환경에 적용시킬 수 있는 장점이 있으나, 여러 곳에 분산되어 있는 서비스를 효율적으로 사용할 수 있는 CORBA의 장점을 활용 할 수 없는 단점이 있다.

3. 분산 객체 프레임웍을 지원하기 위한 재공학 시스템

본 논문에서는 객체지향 언어인 C++로 이루어진 독립적인 시스템이나 클라이언트/서버 환경에서 동작하고 있는 시스템을 분산객체 프레임웍인 CORBA환경에서 동작할 수 있도록 재공학 하는 방법을 제안한다. 이를 위해 기존의 객체 지향 레거시 시스템을 역공학을 통하여 추상적인 형태로 바꾸고, CORBA 제약 조건에 맞도록 재구성하여 새로운 컴포넌트들의 집합으로 만들어야 한다. 그림 1은 C++ 레거시 시스템을 분산객체 프레임웍인 CORBA 환경으로 이주(migration)하는 전체 시스템의 구조를 간략하게 나타낸다.

4. 역공학 서브시스템

역공학 서브시스템(그림 2)는 C++ 클래스 분석기와 컴포넌트 변환기로 구성된다. 이 서브시스템은 C++ 레거시 시스템의 소스코

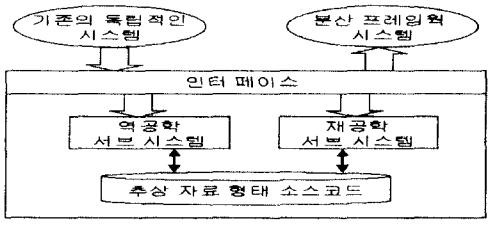
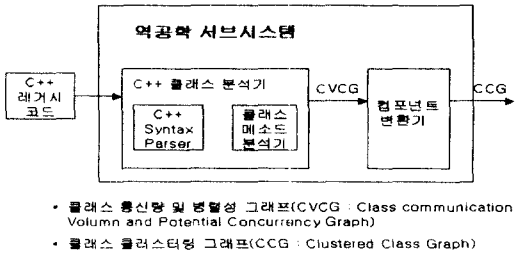


그림 1. 시스템 구조



- 클래스 통신량 및 병렬성 그래프(CVCG : Class communication Volumn and Potential Concurrency Graph)
- 클래스 클러스터링 그래프(CCG : Clustered Class Graph)

그림 2. 역공학 서브시스템

드를 입력 받아 파싱하여(parsing) 클래스 통신량 및 병렬성 그래프(CVCG : Class communication Volumn and Concurrency Graph)를 생성하고, CVCG에서 도출된 클래스간의 통신량과 병렬성을 척도로 관련 클래스끼리 클러스터링 하여 전체 시스템을 몇 개의 서브시스템의 집합으로 나눈다.

4.1 C++ 클래스 분석기

C++ 클래스 분석기는 C++ syntax parser와 C++ 클래스 메소드 호출관계 분석기로 이루어 진다. C++ syntax parser는 각 클래스를 식별하고 코드 및 구문 분석을 통해 CVCG를 생성한다.

CVCG는 노드(node)와 간선(edge)로 구성된다. 노드는 각각의 클래스를 나타내고, 간선은 노드와 노드사이를 연결하여 두 노드 사이의 통신량과 병렬성을 나타낸다.

C++ 클래스 메소드 호출 관계 분석기는 클래스간의 통신량을 측정하기 위해 메소드 간의 호출관계를 분석한다.

표 1. 클래스 통신량 및 병렬성 그래프

+ CVCG = (N, E)
+ E = (V, C)

N : 클래스(node)
E : edge
V : 클래스간의 통신량
C : 클래스간의 잠재적 병렬성

4.2 컴포넌트 변환기

컴포넌트 변환기는 C++ 클래스 분석기의 산출물인 CVCG를 입력받아, 병렬성(C)과 통신량(V)을 척도로 하여 관련 클래스들을 클러스터링 하여 추상표현형태인 클래스 클러스터링 그래프(CCG: Clustered Class Graph)로 정보저장소에 저장한다.

객체 클러스터링에는 크게 세가지 접근법 - Graph-theoretic, 0-1 정수 프로그래밍, 휴리스틱 방법 - 이 있다. 이 중 실세계 문제를 다루는 데, 휴리스틱 접근법이 실용적이다[8].

따라서, 클래스를 클러스터링 할 때, 다음 기준에 따른 휴리스틱 알고리즘을 적용한다.

가) 클래스 간의 통신량은 줄이고,
나) 객체간의 잠재적인 병렬성은 높인다

클래스 클러스터링과 컴포넌트 할당의 목적은 전체 소프트웨어

시스템의 실행시간을 줄이고, 시스템의 신뢰성을 높이는 것이다. 클래스 클러스터링 알고리즘은 분산 소프트웨어 시스템의 성능에 직접적인 영향을 주기 때문에, 전체 시스템의 실행 시간을 줄이기 위해 객체들 사이에 잠재되어 있는 병렬성을 찾아내고, 모듈간의 전체 통신량을 줄일 수 있어야 한다. 모듈간의 통신량을 줄이기 위해서는 생성 모듈 수가 가능한 적게 클러스터링 해야 하고, 객체간의 병렬성을 높이기 위해서는 가능한 많은 모듈로 나누어야 한다. 따라서, 클러스터링 알고리즘은 두 가지의 서로 상반되는 조건을 동시에 만족하는 모듈들로 클러스터링 해야 한다.

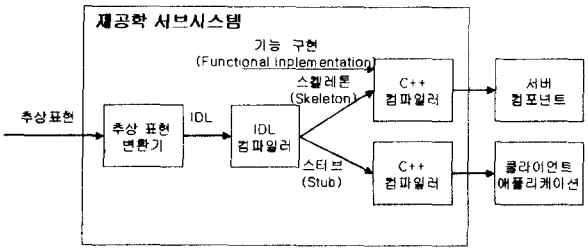


그림 3. 재공학 서브시스템

5. 재공학 서브시스템

재공학 서브시스템 (그림 3)은 추상표현 변환기(Abstract Representation Translator), OMG의 IDL to C++ 컴파일러, 그리고, 기존의 C++ 컴파일러로 구성된다. 이 서브시스템은 역공학 서브시스템에 의해 나누어진 몇 개의 서브시스템의 추상 표현을 입력받아 CORBA기술을 사용하여 분산 시스템을 생성한다.

5.1. 추상 표현 변환기

추상 표현 변환기(그림 4)는 역공학 서브시스템에서 추출한 정보로 클러스터링한 컴포넌트의 집합을 추상 표현으로 기술하고, 이 추상 표현을 입력받아 OMG의 IDL로 매핑한다.

추상 표현 자료 구조를 C++ 언어와 유사하게 정의하여, 기존의 C++ 과 IDL의 1:1 매핑 규칙 테이블을 이용 할 수 있도록 한다.

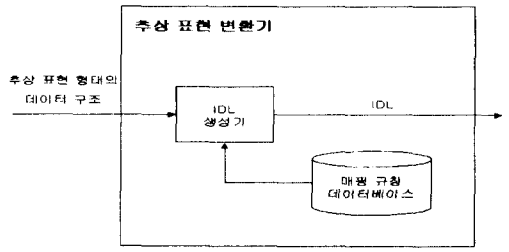


그림 4. 추상 표현 변환기

5.2. IDL 컴파일러와 C++ 컴파일러

IDL 컴파일러는 Visibroker의 IDL 컴파일러(idl2cpp)를 사용하여 추상 표현 변환기가 생성한 IDL로부터 Server skeleton과 Client Stub를 만들어 낸다.

C++ 컴파일러는 IDL 컴파일러가 생성한 skeleton과 stub로 서버와 클라이언트를 구현한다.

6. 결론 및 향후 연구

본 논문에서는 C++과 같은 객체 지향 언어로 이루어진 독립적인 시스템이나 클라이언트/서버 환경에서 동작하고 있는 시스템을 CORBA와 같은 분산 객체 환경에서 동작할 수 있도록 재공학 하는 시스템을 설계하였다.

이 시스템은 레거시 시스템으로부터 소스코드를 입력 받아 역공학 서브시스템을 거쳐 병렬성을 높이고, 통신량을 낮추는 방향으로 관련 클래스들을 클러스터링하여 정보저장소에 추상표현형태로 저장한다. 추상표현형태로 만들어진 자료는 재공학 서브시스템을 거쳐 CORBA환경에 맞게 분산 시스템으로 재공학 한다.

정의된 시스템을 이용하여 레거시 시스템을 재공학 하면 신뢰성과 상호 운용성을 높이고, 프레임웍 기반의 효율적인 소프트웨어 개발을 효과적으로 지원 할 수 있다.

현재 이 시스템의 시범도구(Prototype) 구현하고 있는 단계에 있으며, 향후 추상표현형태가 정보저장소에 정의되는 형식과 통신량, 병렬성을 측정하는 방법을 정형적(formal)으로 정의에 관한 연구를 추진할 계획이다.

7. 참고문헌

[1] Y. F. Chen, E. R. Gansner, and E. Koutsofios, "A C++ Data Model Supporting Reachability Analysis and Dead Code Detection," Proceedings of the 6th European Software Engineering Conference, 1997.

[2] Daniel Jackson and Eugene J. Rollins, "A New Model of Program Dependences for Reverse Engineering," Proceeding of the ACM SIGSOFT 94 Symposium on the Foundations of Software Engineering, pp. 2-10, 1994.

[3] K. Koskimies, H. Mossenbock, "Scene: Using Scenario Diagrams and Active Text for Illustrating Object-oriented Programs," Proceedings of ICSE-18, 1996.

[4] C. H. Kung, J. Gao, P. Hsia, J. Lin, and Y. Toyoshima, "Design recovery for software Testing of Object-Oriented Programs," Proceeding of the Working Conference on Reverse Engineering, 1993.

[5] John Parodi, "Building Wrapper for Legacy Software Applications," http://www.digital.com/info/objectbroker/product/obwp_wrap.htm, Dec. 1997

[6] Stephen S. Yau and Haiqing Ying, "A Clustering Algorithm for Object-Oriented Development of Distributed Computing System Software," Proceedings of the Fifth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems, pp.274-281, 1995.