설계 정보 복구와 객체 지향 구조의 논리적 분석을 통한 재구성 툴 설계 및 구현

김 행 곤[†] · 최 하 정[††] · 변 상 용[†††] · 정 연 기[††††]

요 약

소프트웨어 재공학은 기존 시스템의 새로운 기법들과 소프트웨어 툴의 유지보수를 지원함으로써 기존 시스템을 증진하고 소프트웨어 유지보수성을 향상시키는데 적용된다. 소프트웨어 재공학은 일반적으로 기존 시스템의 소프트웨어 컴퍼넌트를 추출하고 기존 시스템을 이해하는데 도움을 준다. 본 논문에서는 재공학을 위한 프로그램 분석과 재공학 환경을 지원하는 툴을 논한다. 프로그램 분석은 기존 프로그램이 입력으로 제시되면 구조적이고 객체 지향 부분에 대한 정보를 생성한다. 이것은 재공학 방법론을 통해 추출된 코드로 정보에 의해 재구성되어질 수 있다. 이러한 재구성 정보 중 객체 지향 구조로의 정보는 직접 관계와 요약 관계를 통해 질의하도록 프롤로그 형식으로 매핑되어진다.

본 논문에서 설계 구현한 SORS(SOftware Reengineering System)는 (1) 구조적이고 객체 지향 정보를 분석하도록 효과적인 방법론을 수행하고, (2) 기존 프로그램의 복잡성을 제거하며, (3) 재구축을 위한 새로운 코드와 시스템 상호 구조의 재사용 컴퍼넌트들을 조합하고, (4) 프로그램과 제어 구조의 단순화와 같은 기법들을 통해 기존 소프트웨어에 대한 고수준의 이해성과 유지보수성을 제공한다.

The Design and Implementation of Restructuring Tool with Logical Analysis of Object-Oriented Architecture and Design Information Recovery

Haeng Kon Kim[†] · Ha Jung Choi[††] · Sang Yong Byun[†††] · Yun Ki Jeong[††††]

ABSTRACT

Software reengineering involves improving the software maintenance process and improving existing systems by applying new technologies and software tools. Software reengineering can help us understand existing systems and discover software components that are common across systems. In this paper, we discuss the program analysis and environment to assist reengineering. Program analysis takes an existing program as input and generates information about structured part and object-oriented part. It is used to restructure the information by extracting code through reengineering methodology. These restructuring informations with object-oriented architecture are mapping prolog form to query by using direct relation and summary relation.

† 정 회 원:대구효성가톨릭대학교 컴퓨터공학과 부교수
†† 준 회 원:대구효성가톨릭대학교 대학원
††† 정 회 원:제주대학교 정보공학과 조교수
†††† 정 회 원:경북산업대학교 전자계산학과 부교수
논문접수:1996년 1월 30일, 심사완료:1996년 8월 12일

The SORS(SOftware Reengineering System) supports a high-level understandability and maintainability for existing software from the following techniques, (1) by performing effective method to analysis structured and object-oriented information, (2) by erasing code complexity of existing program, (3) by clustering reuse component for new information and system interarchitecture to rebulid, and (4) by simplifying the program and control structure.

1. 서 론

최근 소프트웨어 유지보수와 재공학(Reengineering)을 위한 많은 방법론과 툴들이 제안되어왔다. 그러나 이들은 아직 적용하기에 미비하거나 부족한 점이 많고 여러가지 한계점이 있다. 즉, 문서들이 너무 일반적이거나 너무 상세한 내용들을 많이 가지고 있어서 시스템을 이해하기 위한 초기 접근에서만 유용하였고, 구현단계에서는 무시되어졌으며, 세목들이 너무 많음으로 인해 유지보수자에 대한 지원이 부족하였다. 최근에는 이러한 결점을 보완하도록 요약과 상세화된 문서들 간의 추적성을 가진 소프트웨어 툴들이 제안되어지고, 이들 툴은 그래픽 표현과 그래픽 질의 시스템(Graphic Query System)의 특성을 가지고 있어 유지보수자들이 구현 단계에서 필요로 하는 상세한 정보와 관계를 표현하는 그래프로 나타내어 다중적으로 부분화되어 복합적으로 연결되어졌으나, 아직 미비하여 기초 설계 알고리즘을 그래픽적으로 표현하기에 효과적이지 못하다[1].

또한 소프트웨어 유지보수를 위한 역공학(Reverse engineering)에서도 유지보수자가 요구하는 데이타를 충분히 포함시키지 못할 경우가 발생하는데 이는 소프트웨어 개발을 위해서는 세밀하고 유용할 수 있지만, 유지보수를 위해서는 그다지 충분하지 못한 문제점을 가지고 있다. 그리고, 역공학 툴들은 미리 정의된 보고서들만 생성하고, 정해진 질의들에 대한 답변만을 제공함으로 비유연적이다.

이러한 문제점들을 해결하기 위해서는 데이타 베이스의 구별된 부류에 저장되어야 하는 밀접히 관련된 사실의 집합, 다른 사실들의 상호관계와 부분집합의 추출을 허용하는 질의 언어(Query Language), 그리고 실제적으로 저장된 사실로부터 시작하는 질문에 대한 답변을 정의하는, 일반적이고 요약된 규칙들의 발전된 집합의 특징을 가지는 소프트웨어 시스템의 관점을 표현하도록 환경과의 상호동작이 필요하며 이를 통해 필요한 정보의 추출과 사용자간의 인터페이스를 더 쉽게 수행하도록 한다. 또한 특정한 질의에 대답할 수 있는 툴은 역공학에 의해 생산된 문서의 완전성과 이해성 사이의 계약 관계를 극복할 수 있도록 해주므로, 유지보수자에게 수행되고 있는 특정한 연산을 위해서 복구할 수 있는 정보를 사용자에게 표시할 뿐만 아니라, 그것 모두를 연결하고 사용자의 특정한 요구에 따라 처리할 수 있다[2].

따라서 본 논문에서는 기존 원시코드로부터 추출된 정보를 구조적인 부분(Structured Part)과 객체 지향적인 부분(Object-Oriented Part)으로 나누어, 5단계의 재공학 방법론을 이용하며 구조적이고 객체 지향적인 새로운 재구성된 정보코드 형태로 생성한다. 객체지향 리스트 구조로 재구성(Restructuring)된 정보는 사용자와 질의를 할 수 있도록 직접관계(Direct relation)를 통해 프롤로그 형식으로 매핑한 후, 요약관계(Summary relation)를 사용하여 코드 정적 분석에 의해서 클래스의 정보를 저장하는 SORS(SOftware Reengineering System)를 설계 및 구현한다. SORS를 통해 기존 소프트웨어의 코드 복잡성 감소, 제어흐름의 단순화 및 프로그램 구조 등, 새로운 정보의 재사용(Reuse)과 시스템의 내부구조 및 흐름들을 보여줌으로써 사용자와의 질의를 통해 이해 가능케 하고 나아가 유지 보수의 효율성을 증대시키는데 주력한다.

2. 기존 재공학 툴

대부분의 기존 역공학 툴은 두가지 범주에 속하게 된다. 첫째, 역공학과 재문서화(Redocumentation) 툴 부분으로, 제어흐름이나 데이타 흐름처럼 프로그램 이해와 메뉴얼 재공학을 돕기 위해서 프로그램의 구조를 다른 시각으로 표현한다. 둘째, 이외의 툴들은 프로그램에서 GOTO를 제거하는 것처럼, 하나의 언어에서 또다른 언어로의 자동적 번환 혹은 자동화된 재구성을 제공한다. 이러한 툴들은 사용자와의 상호

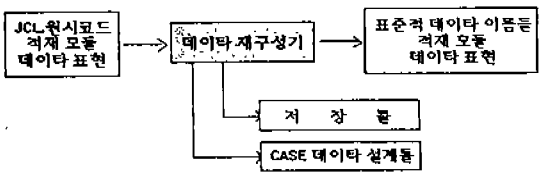
작용에 대한 요구가 거의 제시되지 않으며 설계 정보 복구(Design information recovery) 혹은 유지보수에 대해서도 언급하지 않는다.

이러한 접근방법들은 역공학, 재문서화, 그리고 저단계 프로그램 엔티티(entity)의 번역을 제공하며, 또한 새 모듈로의 데이타와 프로그램 문장들을 재구성하도록 자동화된 분석과 사용자 상호작용의 조합으로 제공된다[4].

2.1 데이타 재구성툴

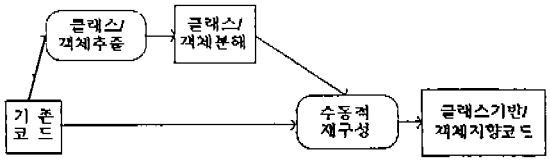
프로그램 논리 재구성과 데이타 재구성은 시스템간의 데이타 정의들을 표준화하는 과정으로 이들의 이해성 증진에 의해 기존 시스템의 유지보수성이 증진된다. 이들은 프로그램 논리와 데이타를 표준화하는 과정으로, 데이타 재구성은 시스템 단계의 재구성 형태를 지니며 이해성을 최적화하기 위해서는 시스템에서 사용된 데이타들을 항상 같은 방법으로 정의되어야 하고, 다른 시스템 간에서도 같은 이름으로 사용되어야 한다. 또한 이들은 사전과 저장소, 데이타 모델링, 데이타베이스 관리 시스템, 소프트웨어 재사용성과 같은 새로운 기법들로의 이주에 관해 수행된다.

데이타 재구성툴들은 데이타 구조를 분석·수정하고 데이타 구성요소 추적과 함수 수정으로 비일치적이고 의미없는 데이타 요소명들을 표준화하고 의미있는 데이타 구조로 전환하는 유지보수기이다. 같은 이름으로 추출하고 같은 방법으로 정의된 시스템에서 사용하는 모든 데이타를 확인하도록 시스템과 프로그램간에 정의하여 어느 곳에서도 사용할 수 있고 새로이 형식을 이룬 표준 이름들과 정의와 다른 속성들을 가진 통괄적인 정보저장소를 이용한다. 데이타 재구성툴은 (그림 1)과 같이 나타낼 수 있다[3].



(그림 1) 데이타 재구성기
(Fig. 1) Data Restructuring Tool

2.2 반자동 객체 추출기



(그림 2) 반자동 객체 추출기
(Fig. 2) Semi-Automatic object extractor

기존 코드를 객체 모듈 형태로 먼저 재구성하는 방법이 Zimmer[5]에 의해 시도되어 Jacobson[6]에 의해 기존 시스템을 수동적이고 점진적으로 객체지향 시스템으로 재공학하는 방법을 제시하였다. Fieldman [7]은 (그림 2)에서 제시한 반자동 객체 추출기를 제안하였는데, 기존Fortran 프로그램을 반자동적으로 C 혹은 C++로 전환하기 위해 클래스/객체를 추출하는 방법을 언급하였다. 그러나, 이 방법은 아직 객체를 인식하거나 생성할 수 없으므로 객체 지향 코드를 생성하지 못한다. 이외에 재구성에 대해 연구한 내용들은 다수가 있으나 아직은 미비한 상태라 할 수 있다 [8][9][10].

2.3 LaSSIE

LaSSIE의 목표는 복잡한 응용의 설계를 설명하고 재사용에 관련된 코드를 식별하며 도메인 모델, 설계 추상화, 특징 추적, 코드 추적을 저장하기 위해 프레임 기반 전문가 시스템을 사용하는데, 전문가 시스템은 프로그램 지식 베이스와 데이타 베이스 모델링에 대해 많은 중요한 기능을 제공한다. 그러나, 설계 추상화가 파서에 의해 계산되어지는 동안 수작업에 의해서 도메인 모델이 구성되는 단점이 있다[11][12].

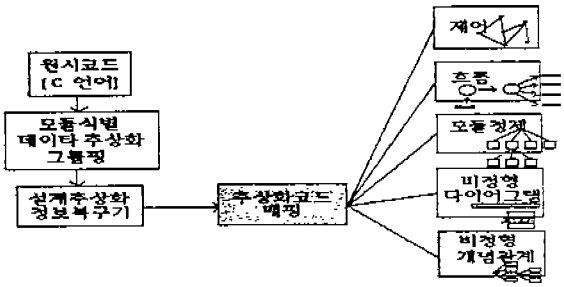
3. 소프트웨어 재공학 시스템 SORS(SOftware Reengineering System) 설계

3.1 재공학 방법론

본 논문에서는 설계 정보 복구와 객체 지향 구조로의 재공학 방법론을 적용시켜 기존 원시 코드로부터 구조적이고 객체 지향적인 정보를 추출해내도록 하며, 객체 지향적으로 재공학된 정보는 프롤로그적인

단계 1 : 제공된 프로그램 이해
단계 2 : 객체 추출
단계 3 : 소프트웨어 재사용과 복구 라이브러리 제공
단계 4 : 제설계 및 재구성
단계 5 : 제설계 · 재구성 정보의 논리적 관계 추출

(그림 3) 재공학 방법론 단계
(Fig. 3) Reengineering Methodology step



(그림 4) 기본 설계 복구 과정
(Fig. 4) The basic design recovery process

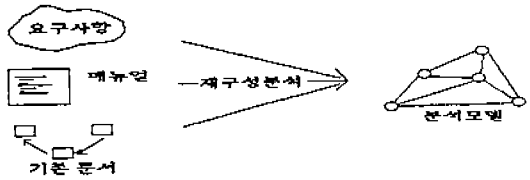
접근을 통해 사용자와 질의를 할 수 있도록 하는 5단계의 재공학 방법론을 제시한다(그림 3)[13][14].

단계 1:제공된 프로그램 이해(그림 4).

이 단계에서는 입력된 원시 C 프로그램의 이해를 돕는 과정으로, 먼저 서브 시스템의 구조, 모듈 구조, 중요한 데이타 구조와 같은 큰 규모의 기본 구조를 살핀 다음, 다양하고 유용한 설계 구조를 복구하여 비정형 다이어그램, 비정형 개념과 관계, 모듈 정제, 흐름, 제어와 같은 추상화된 형태로 이들을 나타내며, 다양한 추상화와 이들을 구현하는 코드 세그먼트간의 관계를 다룬다.

추상화 정보 복구기는 원시코드로 부터 받아들여진 비정규적 다이어그램을 정규적 개념과 관계, 설계 이론, 모듈 정제와 데이타 제어의 흐름으로 대응할 수 있도록 복구시키며, 추상화 코드 매핑 정보를 통해 이러한 관계를 복구·정규화한다.

시스템의 분석모델을 정의할 때는 시스템에 대한 기초 정보와 일치해야 한다. 기존 정보는 요구사양 명세서, 사용자 운영 지침서, 유지보수 메뉴얼, 교육 메뉴얼, 설계 문서, 소스 코드 화일, 데이타베이스 스키마 기술등을 가진다. 기술요소의 중요한 부시스템은 시스템을 표현하는 기술 요소의 집합이며, 원시코드 혹은 원시 코드와 일치하는 문서라고 할 수 있다. 시스템의 분석 모델은 기존 문서, 메뉴얼 그리고 요구사양을 통해 재구성하여 얻은 정보이며 이는 하나의 그래프로 생각할 수 있다(그림 5).



(그림 5) 분석모델
(Fig. 5) Analysis Model

단계 2:객체 추출[25]

객체와 클래스 추출시에는 객체의 인스턴스 변수를 추출하는 부분과 메소드를 추출하는 부분으로 나누어 얻을 수 있다.

① 국부 객체 추출:객체 데이타 구조는 국부적으로 정의되며, 인스턴스 변수를 다루는 문장들도 코드로 얽혀 있어 인스턴스 변수들과 메소드의 구분이 명확하지 않으며, 변수들이 국부적으로만 사용되기 때문에 부호의 지속성을 개선하는데 크게 기여하지 못한다.

② 전역 객체 추출:구조체와 공통의 자료 구조, 그리고 단위 루틴을 이용하여 전역 변수들을 하나의 객체로 추출하여 객체의 속성이 되고, 단위 루틴들은 전역 변수들 중 동작의 중심이 되는 메소드가 된다.

③ 매개 변수에서의 객체 추출:객체의 구성 부분이 될 수 없는 매개변수도 고려하여, 얼마나 많은 객체가 형식적 리스트로 형성되는가를 결정하고, use-only, use-and-set, define-only 로 분류한 후 가장 큰 범주에 있는 변수들을 한 객체의 인스턴스 변수로 만들며, 만약 인스턴스 변수가 use-only이면 한 객체의 상태를 읽는 방법을 추출해내고, define-only이면 하나의 객체를 생성해

내는 생성자를 추출하게 된다. 그 외에는 객체의 상태를 수정해 주는 방법을 얻게 된다. 이러한 분석에서 기능의 이름은 호출자에게 반환값을 돌려주는데 사용되므로 형식적 매개 변수로 취급된다.

④메소드 추출:클래스의 객체가 수정되는 기능을 제공해야 하며, 한 클래스의 인스턴스 변수들을 알면 그 클래스의 메소드를 추출하고, 하위 프로그램에 있는 데이타 흐름을 분석, 분리하여 메소드를 추출한다. 한 블럭의 문장들은 둘 또는 그 이상의 공통 블럭들에서도 변수를 사용할 수 있으며, 그런 경우 둘 또는 그 이상의 메소드가 만들어질 수 있고, 같은 공통블럭을 사용할 수 있으므로, 이러한 루틴들에서 찾을 수 있는 메소드는 결합해 주어야 한다.

단계 3:소프트웨어 재사용과 복구 라이브러리 제공

복구된 설계 구성요소의 사용에 관한 과정으로서, 이들의 재사용 향상에 관한 구성요소의 생성에 주력한다. 일반화는 광대역 응용에 대해 적용가능한 구성요소를 만들지만, 이것은 독립적인 설계가 중복되지 않도록 이들에 대한 분해를 요구할 수 있다. 또한 재사용 라이브러리와 도메인 모델내에서 새로운 추상화로 통합된다. 따라서 더 단순한 새 구성요소 구축과 다른 시스템으로부터 더 단순한 요소의 복구를 통해 그 정보의 재사용을 기대할 수 있으며 이들간의 관계를 (그림 6)과 같이 나타나고 있다.

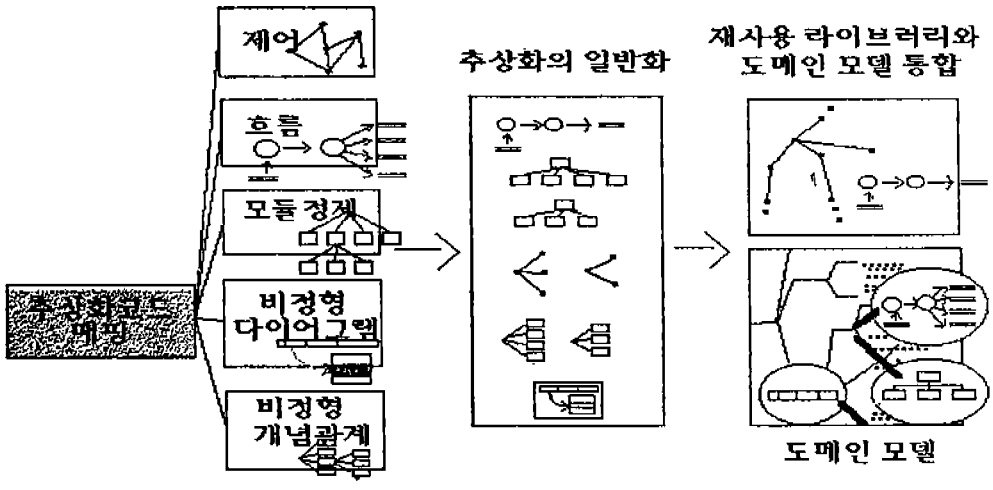
단계 4:재설계 및 재구성

재설계와 재구성은 프로그램 규칙과 객체모델로의 변형을 위한 추상화 코드 매핑을 기반으로 이루어지는데, 논리적 분석 문서는 프로그램의 논리적 흐름 이해를 위해 사용되는 문서이며 크게 물리적 분석, 논리적 분석 단계를 통해 획득된다.

물리적 분석문서는 모듈과 관련하여 구조도 제어 흐름도 등의 다이어그램을 포함하며, 논리적 분석 문서에는 재공학과 관련된 유용한 정보를 가진다. 논리적 분석 문서와 물리적 분석 문서의 획득으로 이해성과 유지보수성을 향상시킨다(그림 7).

구조적 프로그래밍이란 요구된 정확한 스타일 형식(모듈러, 계층적 제어구조)의 규칙 집합 논리 구성의 제컴파일된 집합에 따라 프로그램을 구성하는 방법이며, 추상화 코드 매핑은 기존의 C 코드로부터 객체 클래스화할 수 있는 정보를 모아 추상화하도록 C++으로의 표현을 말한다.

단계 5:재설계·재구성 정보의 논리적 관계 추출 [15][16]



(그림 6) 소프트웨어 재사용 라이브러리로 확장

(Fig. 6) Design recovery extensions supporting reuse library

소프트웨어 유지보수의 향상을 위해서 논리적인 분석 단계를 통해 사용자와의 인터페이스를 이룰 수 있도록 프롤로그의 접근을 행한다. 추출되어질 정보들은 클래스 데이타 멤버 종류와 내용, 클래스 함수 멤버의 내용, 클래스에 호출되는 메소드, 클래스 상속과 같은 내용으로 질의를 행하도록 한다.

프롤로그의 접근 중 맨처음 단계가 직접관계로, 같은 클래스에 속하는 정보간에 존재하는 기본 관계를 개별적으로 분석가능토록 애트리뷰트, 메소드, 파라미터 등과 같은 정보를 코드로부터 쉽게 얻을 수 있다. 요약관계는 이러한 직접관계를 바탕으로, 데이타 가시성 문제를 해결하는 부분으로서 다른 클래스와의 상호 작용에 대한 규칙이다. 직접관계에서 추출한 규칙들을 기반으로 전개할 객체 정보추상화를 제공하도록 class_dec_scope과 attribute_or_par_dec을 선언하여 가시성 해결의 기초를 마련한다. 먼저, 데이타 요소 x를 변수와 형식 매개변수 양쪽 모두로 선언하고 있는 클래스 m에서 보여주는 것과, 두번째로 클래스 m2에서 선언된 데이타 요소 x는 만약 m1이 x를 재선언하지 않고 m1을 직·간접적으로 선언하는 m2에 직·간접적인 변수가 없다면, 변수 혹은 형식 매개변수를 m1에서 보여준다.

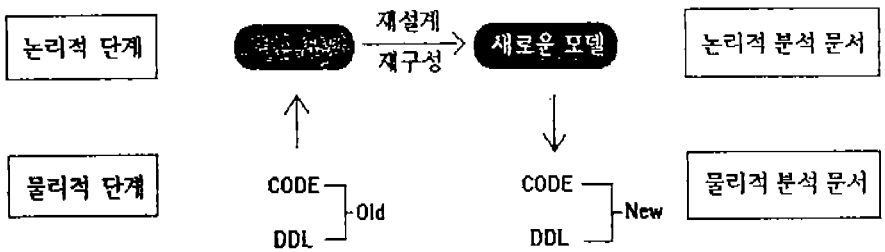
즉, 이것은 m2에서 선언된 데이타 요소 x가 m1에서 보여진다는 것은 (m2=m1) 혹은 ((m1, x)∈var_or_par_dec AND (m_i, x)∈var_or_par_dec)가 (m2, m_i)∈mod_dec_scope AND (m_i, m1)∈mod_dec_scope을 모든 m에 대해 만족함을 나타낸다.

(그림 8)에서는 상위 추상화에 대한 내용으로서 하위 요약관계를 토대로 클래스의 상속성에 대한 사항을 직접 질의 가능토록 적용할 수 있다. 이것은 프롤로그를 기반으로한 사용자와의 질의응답식의 질문을 행하여 실제 클래스의 상속성에 관한 사항과 클래스의 계층적인 부분을 살펴볼 수 있으며 이는 실행 예제 부분에서 다루도록 한다. (그림 8)의 프롤로그 규칙은 클래스내에서 멤버의 속성 파악을 위한 것으로서 access_privat(B, E)와 access_protect(B, E), access_public(B, E)가 있으며, accessable(B, E)를 사용하여 클래스 내에서 멤버의 속성 접근의 여부를 결정한다. 또한, 클래스의 상속관계에서 최종 얻고자 하는 정보를 위한 규칙으로는 visible로, visible(C, (x, main))은 클래스 main에서 정의된 데이타 멤버가 클래스 안에서 가시적일때 클래스를 반환하며, visible(C, (A,main))은 클래스 main에서 정의된 함수멤버가 클래스 안에서 가시적일 때 클래스를 반환하도록 한다.

3.2 시스템 구성

SORS의 시스템 구조는 (그림 9)과 같이 추출기 서브시스템과 추상화기 서브시스템으로 나누는데, 추출기 서브시스템은 C 프로그램 데이타 구조를 설계 복구 형식과 C++ 클래스간 데이타 구조로 재구성하고 새로이 재구성된 클래스 정보를 프롤로그적 형식의 매핑을 위해 필요한 직접 관계들을 생산하여 SORS 규칙 정보 저장소(DB)에 저장하는 시스템이고, 추상화기 서브시스템은 요약 관계를 통해서 추상화 과정을 수행하며, 클래스와 클래스간 정보흐름 분석의 결과들을 사용자의 질의를 통해 나타낸다. (그림 10)에서는 파서를 통해 C 원시코드로부터 추출할 수 있는 유지보수 가능한 정보들을 나타내었다.

3.2.1 추출기 서브 시스템



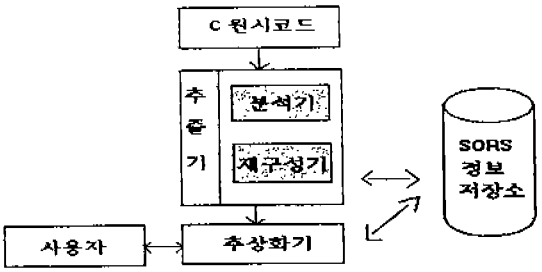
(그림 7) 재설계 및 재구성과정
(Fig. 7) Process of Redesign and Restructuring

accessable:(m,E)∈access_privat,(m,E)∈access_public,(m,E)∈access_protect (m,E)를 만족.

visible(C,(x,B)):클래스 B에서 정의된 데이타 멤버 x가 클래스 안에서 가시적이면 클래스 반환.

visible(C,(A,B)):클래스 B에서 정의된 함수 멤버 A가 클래스 안에서 가시적이면 클래스 반환.

(그림 8) 클래스 요약 관계 규칙
(Fig. 8) Summary Relation Rule of class



(그림 9) SORS 구조
(Fig. 9) SORS structure

추출기 서브시스템은 기존 프로그램 구조와 데이타 흐름을 재구성하기 위해 필요한 직접 관계들을 생산하고, SORS 정보 저장소에 이들을 저장한다.

서브 시스템은 분석기와 재구성기로 구성되는데, 분석기는 C 원시화일을 받아 파스 트리를 생성 및 처리하여 데이타 구조에 대한 정보를 얻어내고, 얻어진 정보는 재구성기를 통해 설계 정보복구에 대한 정보 구조와 C++의 클래스 정보 데이타 구조로 재구성하며, 이를 통해 얻어진 정보 구조들은 프롤로그 규칙들로의 매핑으로 사용된다.

이 시스템은 비구조적이고, 비객체 지향적인 정보를 구조적이고 객체 지향적인 정보로의 재공학 방법론을 조합한 과정이다. 재설계-재구성된 정보는 (그림 11)와 같은 객체정보 리스트구조로 저장되어 직접 관계를 유도해 내며, 이는 클래스 내부 및 클래스간의 정보 분석을 위한 추상화기에 제공된다.

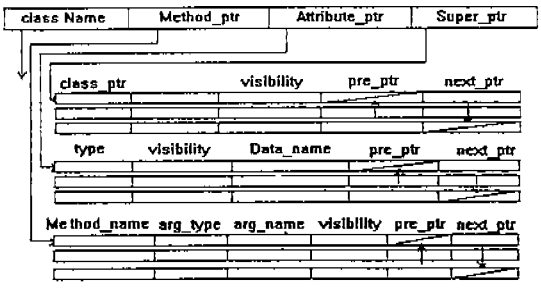
3.2.2 추상화기 서브 시스템

추상화기 서브 시스템은 사용자 질의들을 받고, 모아진 직접 관계들에서 추상화 과정을 수행하며, 클래스 구조와 클래스간 데이타 흐름 분석의 결과를 표현한다. 직접관계를 통해 추상화기 서브 시스템에서는 다른 클래스에서 참조되는 전역변수의 선언을 가진 클래스 이름, 클래스에서 참조되는 형식 매개변수에 적재된 변수들과 관련하여 선언된 클래스들의 리스트, 클래스에서 참조되는 변수에 대한 종속적인 연결 내용을 토대로 관계들의 정확성과 완전성을 추상화할 수 있는 요약관계를 제안한다.

또한 호출과 비호출 클래스의 집합을 위한 클래스

변수 정보 : 선언하고 있는 변수 정보
모듈 정보 : 직접적으로 호출한 모듈 정보
파라미터 정보 : 이름과 위치에 의해 식별되는 파라미터 정보
함수 정보 : 모듈 내에서 사용된 함수 정보

(그림 10) C 모듈 정보 정의
(Fig. 10) Definition of C module information



(그림 11) 클래스 정보구조
(Fig. 11) Structure of Class information

정보 흐름 생성에 대한 문제는 상속성에 관한 문제로 해결하는데, 피호출 클래스 혹은 그의 서브 계층에서 사용되거나 호출 클래스 혹은 그의 상위 클래스들에 의해 선언되는 실변수, 그리고 피호출 클래스와 서브 계층에 의해 정의되며, 호출 클래스 혹은 그의 상위 클래스에 의해서 선언되는 실변수들에 대한 정보를 고려하여 객체 지향의 추상화를 해결한다.

(그림 12)에서의 질의는 (그림 8)에서의 규칙을 질의한 내용으로, 클래스내의 데이타 멤버와 함수 멤버들의 속성 파악을 위한 질의로서 가시성에 대한 내용을 추론한 것이다.

```
accessable(B,E):- access_priva(B,E) or
                  access_protect(B,E) or
                  access_public(B,E).
visible(C,(A,B)):- method_dec(B,A),
                   class_dec_scope(B,C),
                   accessable(C,A).
visible(C,(x,B)):- method_dec(B,x),
                   class_dec_scope(x,C),
                   accessable(C,x).
```

(그림 12) 클래스 가시성 질의(Ⅱ)
(Fig. 12) Class visaal query(Ⅱ)

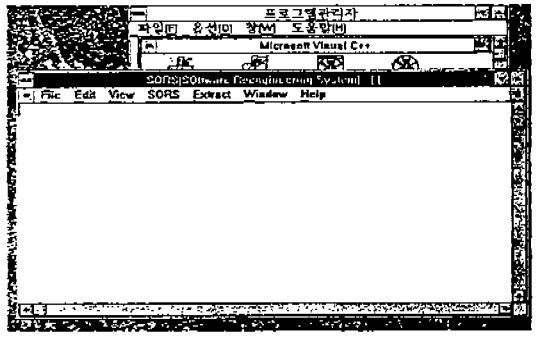
4. 구현 및 평가

4.1 실행 예

정보 추출을 위한 몇 가지 질의를 통해 클래스 정보를 유추하는 SORS의 구현된 실행 예를 제시한다. 초기 화면은 (그림 13)에서처럼 작업환경을 "File", "Edit", "View", "SORS", "Extract", "Window", "Help"의 기능으로 구성되며 이것은 Microsoft사의 Window상에서 Visual C++로 구현된 작업이다.

(그림 14)에서는 제시된 기존 원시 코드로부터 구조적인 형식으로 재공학되어진 결과를 제시한 부분으로, 제시된 원시 코드보다 더 구조적이며 사용자가 이해하기 쉽고 간단하게 표현됨을 알 수 있다.

질의를 통해 사용자와의 인터페이스 부분을 보여주는 부분으로서, 원하는 정보의 내용은 규칙을 토대로 질의하여, 사용된 데이타 멤버와 함수 멤버, 그리고 상속의 관계에 대한 정보를 유추해 나갈 수 있다. (그림 15)에서 질의 결과를 살펴볼 수 있다.

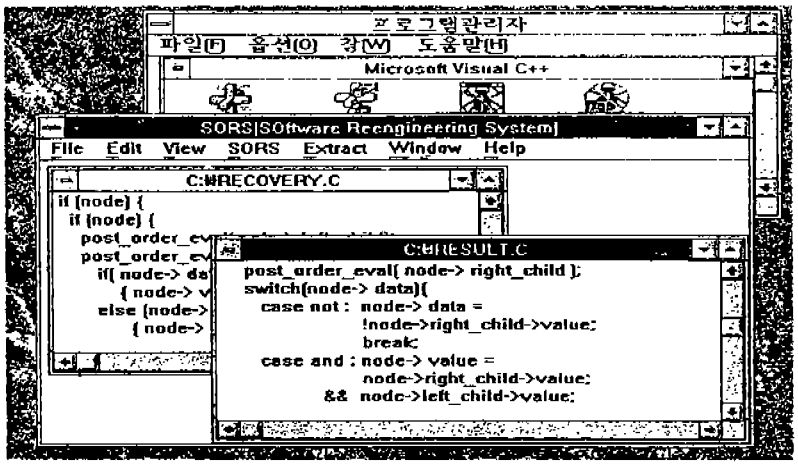


(그림 13) SORS 초기 화면
(Fig. 13) SORS Overview

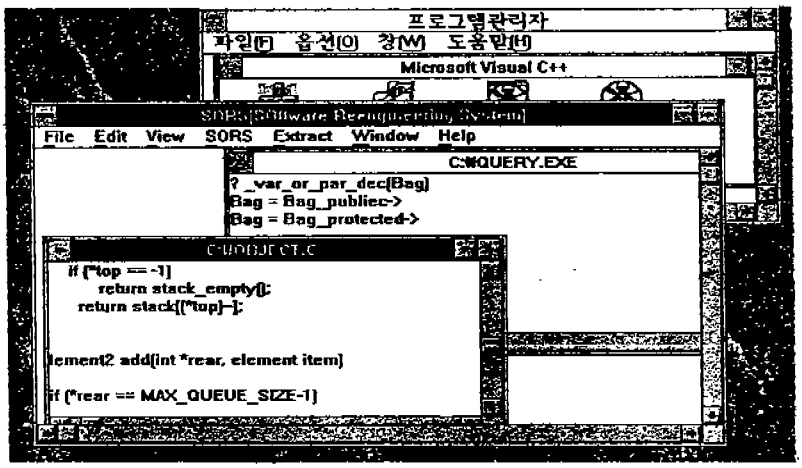
4.2 평 가

본 논문에서 제안한 SORS와 기존에 제시된 여러 연구자들에 의해 제안한 재구성 툴을 비교, 평가한 내용을 다음과 같은 〈표 1〉로 요약할 수 있다.

따라서, SORS는 기존의 구현중이거나 구현된 시스템들과는 달리 비구조적인 면을 구조적으로 재구성하고, 원시코드에서 객체지향적인 부분으로 재공학 가능한 정보를 유추하여 이를 재사용 가능한 라이브러리로 객체를 추출할 수 있도록 하는 자동화된 재공학 시스템으로서, 모든 부분을 C++로 변환하는 것이 아니라, 재사용가능하고 객체지향적일 수 있는 성격의 정보만을 유추하여 객체와 클래스 그리고 메소



(그림 14) SORS 실행 예(Ⅰ)
(Fig. 14) Example of SORS Result(Ⅰ)



(그림 15) SORS 실행 예(Ⅱ)
(Fig. 15) Example of SORS Result(Ⅱ)

〈표 1〉 기존의 타 시스템과의 비교
〈Table 1〉 Comparison of existed other system

연구자	재구성 주요 내용	인식 정도	구현	자동화	논리적 정보 질의	객체 추출
Hauseler 90	COBOL 재구성	객체, 클래스, 메소드 인식안됨	구현중	수동접근	안됨	무
Waters 88	COBOL→HIBOL	객체, 클래스, 메소드 인식안됨	구현됨	수동접근	안됨	무
Ong 93	FORTRAN→C++	자동차 객체, 클래스 인식	구현중	자동변환	안됨	유
Zimmer 90	FORTRAN 재구성	수작업을 통한 객체 인식	구현중	수동접근	안됨	무
Jacobson 91	기존 시스템 재구성	객체, 클래스, 메소드 인식	구현중	수동접근	안됨	무
SORS 96	C→구조적, C++	객체, 클래스, 메소드 인식	구현중	자동변환	가능	유

드로 인식할 수 있으며, 이러한 것을 기반으로 새로운 도메인으로의 재사용 라이브러리화할 수 있도록 한다.

5. 적용예

지금까지 제안한 설계 정보 복구 방법론과 객체지향 구조의 논리적 분석을 이용하여 실제로 적용되어지는 예를 들어보도록 하자. 본 논문의 예제는 비구조적인 면을 담고 있으면서, 객체지향적인 클래스 구조로 재구성되어질 수 있는 원시코드를 제시하여 아주 작은 범위에서 실행가능함을 보여주도록 하였다.

(그림 16)의 예제를 통해 post_order_evaluation에서, 구조적으로 행할 수 있는 부분을 지적하여 재공학할 수 있도록 하며 스택과 큐에서의 저장공간을 하나의 클래스로 이루어 객체 지향적이며 재사용케 하도록 재공학함을 보여준다. 또한 이러한 클래스로 이루어진 내용, 즉 class BAG을 논리적인 분석을 통해 질의할 수 있도록 프롤로그적인 형식의 규칙에 맞게 적용시킴을 보여주었다. 먼저 재공학 단계를 살펴보면 다음과 같다.

1. 재공학 단계

단계 1:유지보수를 위해 제공된 프로그램 이해

이 단계에서는 먼저 유지보수를 위해서 제시된 예제를 통해 얻어질 수 있는 정보를 식별하고 추출하는 단계이다.

(1)모듈 식별 및 데이타 추상화 그룹핑

(그림 16)에서의 왼편 부분을 크게 typedef와 post_order_eval로 나누어 모듈로 식별한 뒤 이를 데이타

Source Code

```
typedef enum {not,and,or,true,false}
                              logical;
typedef struct node *tree_pointer;
typedef struct node {
              tree_pointer left_child;
              logical      data;
              short int    value;
              tree_pointer right_child;
              };

void post_order_eval(tree_pointer node)
{
 if (node) {
  if (node) {
   post_order_eval(node->left_child);
   post_order_eval(node->right_child);
   if (node->data == not)
          { node->value =
          !node->right_child->value;
           }
    else ( node->data == and )
          {  node->value =
          node->right_child->value
 &&       node->left_child->value;
          }
    else ( node->data ==  or )
          {  node->value =
           node->right_child->value
 ||        node->left_child->value;
          }
    else ( node->data == true )
         { node->value = TRUE; }
    else ( node->data == false )
         { node->value = FALSE; }
    }
  }
}
```

```
#define MAX_STACK_SIZE 100
#define MAX_QUEUE_SIZE 100

typedef struct{
      int key;
      } element1;

typedef queue{
         int key;
         } element2;

element1 stack[MAX_STACK_SIZE];
element2 queue[MAX_QUEUE_SIZE];
int top=-1, rear=-1;

element1 delete(int *top)
{
  if (*top ==-1)
      return stack_empty();
  return stack[(*top)--];
}

element2 add(int *rear, element item)
{
  if (*rear == MAX_QUEUE_SIZE-1) {
     queue_full();
     return;
   }
   queue[++*rear] = item;
}
```

(그림 16) C 원시 코드
(Fig. 16) C Source code

추상화하도록 하였다(그림 17).

(그림 17) 모듈 식별 및 데이타 추상화 그룹핑
(Fig. 17) Module identify and data abstract grouping

(2)추상화 매핑

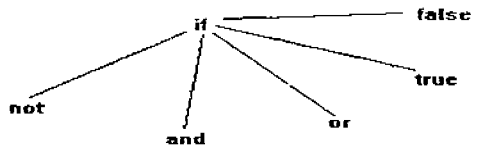
추상화 매핑에서는 이러한 모듈로 묶여진 부분을 제어형태와 흐름 형태, 그리고 모듈 정련 형태로 나누어 추상화기를 통해 매핑하도록 하였다.

①제어형태로 표현

if문을 기준으로하여 not, and, or, true, false로 분기됨을 알 수 있다(그림 18).

②흐름형태로 표현

문장들의 흐름들을 다음과 같은 형태로 표현된다(그림 19).



(그림 18) 제어 형태로의 표현
(Fig. 18) Representation of control style

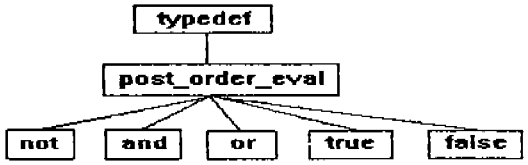
(그림 19) 흐름 형태로의 표현
(Fig. 19) Representation of flow style

③모듈 정련 형태로 표현

모듈화로 나타낸 부분과 흐름에서 표현된 부분을 정련하여 다음과 같이 나타낼 수 있다(그림 20).

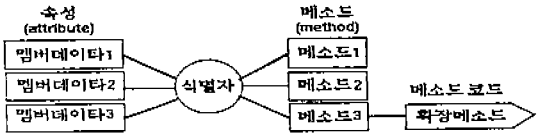
단계 2:분석 모델을 통한 객체 정보 생성

이 단계에서는 추출된 정보를 객체지향적인 클래스 형태로 나타내기 위해서 클래스를 명세한 형식이다. 여기서 제시하는 객체 다이어그램은 기존의 여러



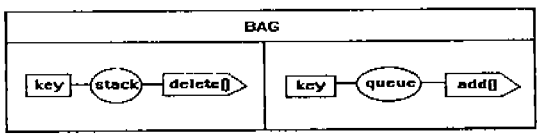
(그림 20) 모듈 정련 형태로의 표현
(Fig. 20) Representation of module refining style

가지 표현방법과 달리 속성부분을 왼편에 제시하고, 메소드 부분을 오른편에 나타내어 객체의 값과 동작을 나타낸다[14](그림 21).



(그림 21) 제안된 객체 다이어그램
(Fig. 21) suggested object diagram

따라서, 위에서 제안된 객체 다이어그램을 이용하여, 본 논문에서 제시된 예제를 적용시켜보면. 다음과 같이 클래스를 기반으로한 식별자 스택과 큐를 추출할 수 있으며, 왼편은 속성인 key, 오른편은 메소드를 표현하여, 각각 delete()와 add()를 나타내며, 이를 Bag에서 지원토록 한다.



(그림 22) Bag을 기반으로한 객체 다이어그램
(Fig. 22) Object diagram based on Class Bag

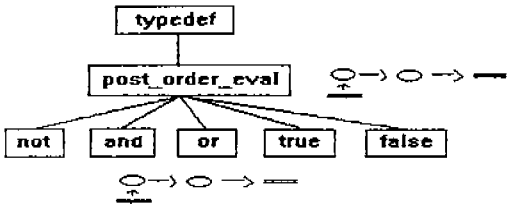
단계 3:소프트웨어 재사용 라이브러리 제공

이 단계에서는 얻어진 정보들을 구조적이고 객체지향적인 형태로 분할하여 재사용 가능토록 라이브러리를 제공하는 단계이다.

①구조적인 형태로의 재사용 컴퍼넌트 구축

앞에서 제시한 3가지 종류의 구조적인 형태를 종합

하여 제시한 결과를 다음과 같이 재사용 가능한 컴퍼넌트로 제시하였다(그림 23).



(그림 23) 구조적 형태로의 재사용 컴퍼넌트 구축
(Fig. 23) Construction of reuse component to structured style

②객체 지향적 형태로의 재사용 컴퍼넌트 구축

Bag이란 클래스를 만들어 Queue와 Stack에서 이를 사용할 수 있도록 구현한 것이다(그림 16).

단계 4:재설계 및 재구성

이 단계에서는 이러한 정보를 재설계 및 재구성하는 단계로서 새로운 코드로 추출할 수 있다.

①구조적으로 복구된 새로운 코드

이것은 구조적인 형태로의 재구성된 예제를 나타낸 것이다(그림 24).

```
typedef enum {not,and,or,true,false} logical;
typedef struct node *tree_pointer;
typedef struct node {
            tree_pointer left_child;
            logical      data;
            short int    value;
            tree_pointer right_child;
            };
void post_order_eval(tree_pointer node)
{
 if (node) {
   if (node) {
    post_order_eval(node->left_child);
    post_order_eval(node->right_child);
    switch(node->data) {
      case not: node->value =
                !node->right_child->value;
                break;
      case and : node->value =
                node->right_child->value &&
                node->left_child->value; break;
      case or:  node->value =
                node->right_child->value ||
                node->left_child->value; break;
      case true: node->value = TRUE; break;
      case false: node->value = FALSE;
    }
   }
}
```

(그림 24) 설계 복구를 통해 재구성된 새로운 코드
(Fig. 24) Restructured new code through design recovery

②객체 지향 구조로 재구성된 새로운 코드

Stack과 Queue의 저장부분을 Bag이라는 공통의 클래스를 사용하여 나타낸 결과로서, 스택과 큐 자체도 각기 클래스를 가지는 형태이며 여기에는 add( )와 delete( )라는 메소드를 가진다(그림 25).

```
class Stack:public Bag
{
 public:
  Stack(int MaxSize=DefalutSize);
  ~Stack();
  int* Delete(int&);
};
Stack::Stack(int MaxStackSize):Bag(MaxStackSize){}
Stack::~Stack(){}
int* Stack::Delete(int& x)
{
  if(IsEmpty()) {Empty(); return();}
  x=array[top--];
  return &x;
}

class Queue:public Bag
{
  public:
  Queue(int MaxSize=DefaultSize);
  ~Queue();
  int* Add(int&);
};
Queue::Queue(int MaxQueueSize):Bag(MaxQueueSize){}
Queue::~Queue(){}
int* Queue:Add(int& y)
{
  if(IsFull())  {Full(); return();}
  array[++rear]=y;
}
```

(그림 25) 객체 지향 구조로의 새로운 코드
(Fig. 25) New code to object-oriented style

여기서는 앞에서 추출된 객체 지향적인 정보를 질의를 통해 사용자가 접근가능토록 하는 단계로서 직접관계와 요약관계를 통한 프롤로그적 형식을 사용한다. 여기서는 (그림 8)에서 정의한 클래스 요약 관계 규칙을 토대로 사용자와 질의하도록 프롤로그를 통해 접근가능토록 하며, 클래스의 계층구조와 상속성 형태, 그리고 추출하고자 하는 클래스 정보들이 어디서, 어떠한 형태로 적재되어 있는 가를 간접적으로 접근할 수있도록 한다.

6. 결 론

기존에 제시된 문서의 미비성과 부족함으로 인한

복잡성, 역공학시 충분치 못한 정보의 제공으로 인해 유지보수를 충분히 해결하지 못하며, 수작업적인 문서를 제공하여 일치성에 문제가 발생하는 경우가 있다. 이러한 여러가지 문제점을 해결하려는 시도는 많이 있었으나 아직 미비한 상태이며 또한 유지보수를 위해서는 프로그램 이해를 돕는 자동화된 툴이 절대적으로 필요하다.

또한, 기능 중심으로 설계 되어진 기존 시스템을 데이타 중심의 객체 지향 시스템으로 재공학하기 위해서 기존 시스템에 잠재해 있는 객체와 클래스를 인식하고 각 클래스의 인스턴스 변수와 메소드를 인식한 다음, 클래스의 상속 구조를 살펴보아야 한다.

따라서, 본 논문에서는 기존 원시코드로부터 추출된 정보를 구조적인 부분과 객체 지향적인 부분으로 나누어 추출한 뒤, 이들 중 객체 지향 리스트 구조로 재구성된 정보는 사용자와 질의를 할 수 있도록 직접 관계를 통해 프롤로그 형식으로 매핑하고 요약관계를 사용하여 객체 정보 흐름을 분석하며 코드 정적 분석에 의한 클래스의 정보를 저장하는 SORS를 설계 및 구현하였다.

이 SORS를 통해 기존 시스템에서의 구조적이고 객체 지향적인 정보를 분석하는 효과적인 방법을 제공함으로써 기존 소프트웨어의 코드 복잡성을 감소시키고 제어흐름의 단순화와 프로그램 구조등, 새로운 정보의 재사용과 원시코드로부터 역공학 툴에 의해 생산되는 정보를 재설계, 재구성함으로써 유추되는 문서정보를 분석·참조하여 프롤로그로 이들의 관계를 표현하며, 시스템의 내부구조 및 흐름들을 특정 질의를 통해 가시적인 관계와 수행 가능적으로 유추할 수 있다.

이러한 SORS는 새로운 도메인으로의 프레임워크를 구축하며 다양한 정보 분석 및 추출을 통한 재사용 컴퍼넌트 데이타 베이스의 구축과, 그리고 제어가능하고 유지보수의 성능을 향상시킨 툴로의 구현 등이 향후 연구 과제이다.

참 고 문 헌

[1] Paul W. oman, "The Book Paradigm for Improved Maintenance," IEEE software, Vol. 12, No. 27, pp. 39-45, Jan. 1990.

[2] Gerardo Canfora, Aniello Cimitile, "A Logic-Based Approach to Reverse Engineering Tools Production," IEEE Trans. on Software Eng., Vol. 18, No. 12, pp. 1053-1064, 1992.

[3] Carma McClure, The Three Rs of Software Automation, Prentice Hall, 1992.

[4] Daniel E, Wilkening, Joseph P. Loyall, "A Reuse Approach to Software Reengineering," J. System Software, Vol, 27, No. 13, pp. 117-125, 1995.

[5] J. A. Zimmer, "Restructuring for style," Software Practice and Experience, Vol. 20(4), Apr. 1990.

[6] Jacoson, I., "Re-engineering of old system to Object-Oriented architecture," OOPSLA '91 Proceeding, 1991.

[7] Feldman, S.J. "A fortran-to-C Converter," Computing science, Technical Report, AT&T Bell Lab, No. 149, 1991.

[8] W, Miller, et al, "Adding Data abstraction to Fortran software," IEEE Software, Vol. 10, No. 30, pp. 278-285, Nov., 1988.

[9] Jonh D. Mcgregre, David A. Sykes, Object-Oriented Software Development: Engineering Software for Reuse, Van Nostrand Reinhold, 1992.

[10] B. Mayer, Object-oriented Software Construction, Englewood cliffs, N.J. Prentice Hall, 1989.

[11] M. Linton, "Implementing relational views of programs," in Proc. of ACM SIGSOFT/SIGPLAN Software Eng. symp. on Practical Software Development Environment, pp. 132-140, 1984.

[12] P. Devanbu, B. Ballard, R. Bachman, and P. Selfridge, LaSSIE: A Knowledge-based software information system. InAutomating Software Design, AAAI/MIT Press, 1991.

[13] 강성구, "기능, 데이타, 인터페이스 분석에 의한 개념적 객체 추출 방법," 포항공과대학 정보산업대학원 석사학위 논문, 1993. 2.

[14] 손이경, "객체지향 시스템의 구성 정보 이해를 위한 툴에 대한 연구," 효성여자대학교 대학 원 석사학위 논문, 1993. 8.

김 행 곤

1985년 중앙대학교 전자계산학과 졸업(학사)
1987년 중앙대학교 대학원 전자계산학과 졸업(이학석사)
1991년 중앙대학교 대학원 전자계산학과 졸업(공학박사)
1978년~1979년 미 항공우주국 객원 연구원
1987년~1990 한국전기통신공사 전임연구원
1988년~1989년 AT&T 객원 연구원
1990년~현재 대구효성가톨릭대학교 컴퓨터공학과 부교수
관심분야: 객체지향시스템 설계, 사용자 인터페이스, 소프트웨어 재공학, 유지보수 자동화툴, CASE

최 하 정

1994년 효성여자대학교 전자계산학과 졸업(학사)
1996년 대구효성가톨릭대학교 전산통계학과 졸업(이학석사)
1996년~현재 경북실업전문대, 안동상지전문대, 계명대 시간강사
관심분야: 소프트웨어 공학(특히, 객체지향 및 재공학)

변 상 용

1982년 중앙대학교 경제학과 졸업(경제학 학사)
1987년 중앙대학교 대학원 전자계산학과 졸업(이학석사)
1991년 중앙대학교 대학원 전자계산학과 졸업(공학박사)
1981년~1983년 코오롱상사 기획실
1991년~현재 제주대 정보공학과 조교수
관심분야: 소프트웨어 설계, 사용자 인터페이스

정 연 기

1982년 영남대학교 공과대학 전자공학과 졸업(공학사)
1984년 영남대학교 대학원 전자공학과 전자계산기 전공(공학석사)
1996년 영남대학교 대학원 전자공학과 전자계산기 전공(공학박사)
1985년~1990년 상지전문대학 전산정보처리과 조교수
1990년~현재 경북산업대학교 공대 전자계산학과 부교수
관심분야: 고속통신망 프로토콜, 멀티미디어 통신, 컴퓨터 구조