韓國 컴퓨터情報學會 論文誌
第9卷 第3號, 2004. 9.

2004-9-3-2-4

규칙기반 역공학 도구의 구현 및 평가

진 영 배*

A implementation and evaluation of Rule-Based Reverse-Engineering Tool

Jin Young Bae*

요 약

소프트웨어의 종류가 다양하고, 커짐에 따라서 유지 보수 문제는 더욱 복잡하고 어렵게 되고, 프로그램 에러의 교정이나 요구 사항의 변경, 사용자에 대한 요구가 점차 증가되면서 소프트웨어의 유지 보수가 소프트웨어 생명 주기에서 가장 비용을 많이 차지하는 부분이 되었다. 본 논문에서는 객체지향 시스템에서 소프트웨어 재구성 환경을 위한 역공학 도구를 설계, 구현하였다. 이 도구는 클래스 정보를 이용한 규칙기반 역공학 도구이다. 프로그램 유지보수자가 프롤로그를 이용한 시스템 쿼리를 사용할 수 있도록 하였다. 대부분의 유사한 클래스를 추출하기위한 재구조화 방법과 변수와 함수 사이의 관계에 근거를 둔 방법을 사용하였다. 추출된 클래스의 가시성이 자동적으로 산출되고, 논리언어를 이용한 쿼리를 사용함으로서 실질적인 유지보수가 이루어지도록 하였다. 또한 이 도구를 평가하는 방법을 제안하였다.

Abstract

With the diversified and enlarged softwares, the issue of software maintenance became more complex and difficult and consequently, the cost of software maintenance took up the highest portion in the software life cycle. We design Reverse Engineering Tool for software restructuring environment to object-oriented system. We design Rule-Based Reverse-Engineering using Class Information. We allow the maintainer to use interactive query by using Prolog language. We use similarity formula, which is based on relationship between variables and functions, in class extraction and restructuring method in order to extract most appropriate class. The visibility of the extracted class can be identified automatically. Also, we allow the maintainer to use query by using logical language. So We can help the practical maintenance. Therefore, The purpose of this paper is to suggest reverse engineering tool and evaluation reverse engineering tool.

▷ Keyword : Rule-Based Reverse Engineering, Class, Abstractor, Prolog Translator, Query

• 제1저자 : 진영배
• 접수일 : 2004.08.18, 심사완료일 : 2004.09.05
* 충청대학 컴퓨터정보학부 부교수

Ⅰ. 서 론

소프트웨어 역공학은 전통적 개발단계 행방향의 역으로 기존 코드로나 데이터로부터 설계사양서나 요구 분석서를 복구시키자는 것이다. 여기에는 두 가지 개념이 있다. 첫째는 코드의 역공학으로 코드로부터 흐름도, 자료 구조도, 자료 흐름도, 등을 역으로 생시키는 것이고, 둘째는 데이터의 역공학으로 자료사전이나 E-R도표 등을 재생시키는 것이다[7]. 그러나 소프트웨어 분석및 설계에 사용되었던 여러 가정들이나 고려되었던 대안들이 코드 안에 내재되어있지 않고 사라져버렸다는 점이다.

기존의 역공학 도구들은 원시 코드를 주로 C,Fortran, cobol등과 같이 비 객체 지향언어를 대상으로하고, 추출하는 정보는 자료에 대한 정보와 모듈 계층에 관한 정보, 프로그램 종속성에 관한 정보로 분류할 수 있다[6][9].

표 1. 역공학 도구
Table. 1 Reverse Engineering Tools

역공학도구	대상언어	결과
ESW	C, FORTRAN	정적분석 문서화기능
CIAS	C	참조, 포함관계
Logiscope	C, PASCAL	정적,동적분석
VIFOR	FORTRAN	호출관계
Rigi Sysyem	C, COBOL	노드관계 통신구조

최근에는 각종 객체 지향 언어가 개발됨에 따라 많은 사람들이 기존 시스템을 객체 지향 언어로 변환하는 방법을 시도하고 있다[2]. 객체 지향 시스템으로의 재구성을 위한 기존의 기법들은 기존에 존재하는 시스템에서 클래스와 객체를 추출해서 클래스에 기반을 두거나 객체 지향 언어를 생성하는 자동 또는 반자동적인 방법을 지향하고 있다. 이와 같이 객체를 자동으로 추출하는 것은 시스템의 변경 비용을 감소시킬 수 있다[1][3]. 이런 코드 번역에 의한 반자동적 접근 방식은 먼저 클래스 객체 추출기가 기존의 시스템에서 가능한 많은 클래스와 객체들을 추출하고, 이것들을 클래스에 기반을 두거나 객체 지향적인 언어로 번역하고 난 후, 클래스와 객체들은 수동으로 새로운 시스템을 만들기 위해 재구성되어진다. 이러한 접근 방식은 전적으로 수동으로 재구성하는 방식에 비하여 시스템 변경 비용을 절감시키고, 코드 개선을 위한 기회를 제공한다는 이점을 가지고 있다[4][5].

본 논문에서는 생성된 클래스 정보를 갖고 사용자가 원래 코드와 이로부터 추출된 클래스 정보로부터 역공학 정보를 얻을 수 있고, 클래스 정보를 완전한 객체 지향 코드로 확장하는 과정에서 유용하게 사용될 수 있도록 한다.

따라서 제한된 형태의 질의가 아니라 다양한 수준의 사용자 질의에 응답할 수 있는 역공학 도구를 구현하기 위해서는 원시정보를 데이터베이스로 구축하고, 사용자 질의에 대해 적절한 응답을 생성해내는 추론 메커니즘이 필요하다. 프롤로그는 술어 논리를 사용한 추론 기능을 제공하므로, 원시정보를 프롤로그 사실로 구축하여 프롤로그에서 제공하는 추론 메커니즘을 통하여 사용자의 질의에 응답하는 방법을 사용한다.

Ⅱ. 규칙기반 역공학

프로그램 재구성을 위한 정보를 추출하기 위해 C 원시 코드와 추출된 클래스 정보[10]를 논리 프로그래밍 언어인 프롤로그로 변환하여 사용자의 질의에 응답할 수 있는 규칙기반 역공학 도구를 설계하였다.

역공학 도구의 구성도는 그림1과 같다. 그림에서 보는 바와 같이 역공학 도구는 프롤로그 변환기와 정보저장소, 정보생성기, 그리고 사용자 인터페이스로 구성된다. 프롤로그 변환기는 원시정보를 프롤로그 사실로 변환하는 모듈이다. 테이블 형태로 구성되어 있는 원시정보에서 추출할 수 있는 직접 관계를 프롤로그 사실로 변환하여 정보저장소에 저장한다.

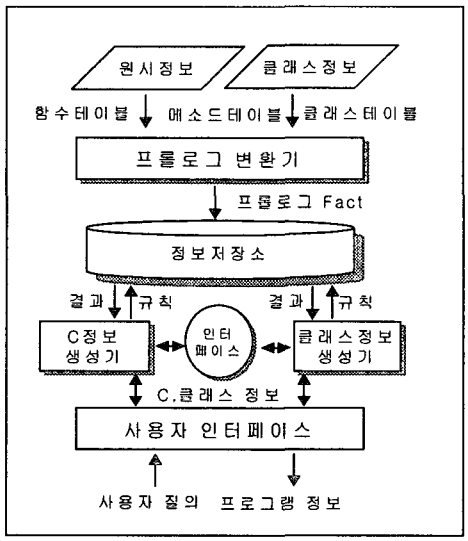

그림 1. 규칙기반 역공학 도구
Fig. 1 Rule-Based Reverse-Engineering Tool

연구방법은 직접 관계변환규칙과 요약관계를 설정하고, 프로그램 유지보수자가 프롤로그를 이용한 시스템 쿼리를 사용할 수 있도록 하였다. 대부분의 유사한 클래스를 추출하기위한 재구조화 방법과 변수와 함수 사이의 관계에 근거를 둔 방법을 사용하였다.

정보생성기에는 직접 관계 사실을 조합하여 얻을 수 있는 추상화된 정보들을 위한 프롤로그 규칙을 미리 정의하여, 이 규칙을 사용한 사용자의 질의에 따른 정보를 제공하도록 구성하였다. 변환기에 의해 프롤로그 사실로 변환된 직접 관계들이 정보저장소에 저장되면 사용자는 인터페이스를 통하여 원하는 역공학 정보를 요구하는 질의를 할 수 있다. 사용자에게 제공되어질 출력 정보를 형식화하는 등의 절차적 처리를 수행하는 규칙들을 정의함으로써 사용자가 다양한 추상화 수준과 형태의 정보를 위한 질의를 수행할 수 있도록 한다. 이러한 정보생성기의 요약 관계 규칙들은 확장 가능하며 새로운 규칙의 정의를 위하여 사용될 수 있다.

2.1 프롤로그 변환기

2.1.1 직접관계 변환 규칙

프롤로그 변환기는 클래스 정보를 직접 관계 생성 규칙에 적용하여 프롤로그 사실을 생성한다. 직접 관계는 클래스 정보로부터 직접적으로 얻을 수 있는 정보들로써 사용자에게 제공되어질 정보들이다. 사용자에게 제공하여야 할 정보는 한 클래스의 정의 정보로부터 여러 클래스간의 상속 관계, 하위 클래스로부터 상위 클래스의 멤버에 대한 가시성 정보 등이 있다.

상속성과 다형성, 그리고 동적 바인딩의 문제는 객체 지향 프로그램의 이해에 있어서 반드시 해결해야 할 것들이고 이를 완벽히 해결하기 위해서는 시스템의 동적 분석이 요구된다. 하지만, 소프트웨어의 동적 분석은 현실적으로 어려운 문제이다. 이를 해결하기 위한 접근법으로 종속성 분석을 통한 클러스터링, 외부 종속성 그래프를 이용한 다형성의 해결, 사용자 협력 시스템 등이 제안되고 있다.

두 개체 사이의 관계가 결정되면 이는 곧 바로 프롤로그 사실로 변환할 수 있다. 다음의 규칙은 직접 관계에 대한 프롤로그 생성 규칙들이다. 추출된 개체 간의 직접 관계들은 곧바로 프롤로그 사실로 변환이 가능하다. 클래스 정보로부터 추출된 직접 관계가 프롤로그 사실로 변환되어 정보저장소에 저장되면 이 사실들에 기반을 두어 여러 가지 질의를 수행할 수 있다.

r1) class(Cn).
정의 : Cn이 클래스 테이블의 첫 번째 열에 존재한다면 Cn ∈ class

r2) var_dec(Cn, Vn)
정의 : Cn ∈ class이고, Vn이 Cn과 같은 행의 두 번째 열이 가리키는 리스트에 존재한다면 (Cn, Vn) ∈ var_dec

r3) method_dec(Cn, Mn)
정의 : Cn ∈ class이고, Mn이 Cn과 같은 행의 세 번째 열이 가리키는 리스트에 존재한다면 (Cn, Mn) ∈ method_dec

r4) public_inherit(Cm, Cn)
정의 : (Cm, Cn) ∈ direct_super이고, Cn이 속해있는 리스트의 유도 타입이 public이면 (Cm, Cn) ∈ public_inherit

r5) public_memV(Cn, Vn)
정의 : (Cn, Vn) ∈ var_dec이고, Vn이 속해있는 리스트의 액세스 타입이 public이면 (Cn, Vn) ∈ public_memV

r6) public_memM(Cn, Mn)
정의 : (Cn, Mn) ∈ method_dec이고, Mn이 속

해있는 리스트의 액세스 타입이 public이면 (Cn, Mn) ∈ public_memM

r7) private_inherit(Cm, Cn)
정의 : (Cm, Cn) ∈ direct_super이고, Cn이 속해있는 리스트의 유도 타입이 private이면 (Cm, Cn) ∈ private_inherit

r8) private_memV(Cn, Vn)
정의 : (Cn, Vn) ∈ var_dec이고, Vn이 속해있는 리스트의 액세스 타입이 private이면 (Cn, Vn) ∈ private_memV

r9) private_memM(Cn, Mn)
정의 : (Cn, Mn) ∈ method_dec이고, Mn이 속해있는 리스트의 액세스 타입이 private이면 (Cn, Mn) ∈ private_memM

r10) protected_memV(Cn, Vn)
정의 : (Cn, Vn) ∈ var_dec이고, Vn이 속해있는 리스트의 액세스 타입이 protected이면 (Cn, Vn) ∈ protected_memV

r11) protected_memM(Cn, Mn)
정의 : (Cn, Mn) ∈ method_dec이고, Mn이 속해있는 리스트의 액세스 타입이 protected이면 (Cn, Mn) ∈ protected_memM

r12) direct_super(Cm, Cn)
정의 : Cm, Cn ∈ class이고, Cn이 Cm과 같은 행의 다섯번째 열이 가리키는 리스트에 존재한다면 (Cm, Cn) ∈ direct_super

2.1.2 요약관계

클래스 정보 테이블은 클래스의 정의, 변수와 메서드의 선언, 액세스 타입, 상속 관계 등을 위한 직접 관계 정보를 가지고 있고, 메서드 테이블은 파라미터 선언과 메서드와 메서드 사이의 메시지 전달 관계를 가지고 있다. 이러한 클래스 정보를 직접 관계 생성 규칙에 적용하면 프롤로그 사실이 생성된다.

이와 같이 원시 프로그램의 직접 관계 정보들이 프롤로그 사실들로 변환이 되면, 사용자는 직접 관계를 사용하여 질의를 할 수 있다.

요약관계는 직접 관계를 적절히 조합하여 얻을 수 있는 관계로서, 직접 관계 사실에 적절한 요약 관계 규칙을 적용함으로써 새로운 사실을 생성할 수 있고, 이 요약 관계를 이용한 질의를 통하여 원하는 정보를 얻을 수 있다. 예를 들어 클래스 상속 계층에서 한 클래스가 다른 클래스의 하위 클래스인가를 알고 싶을 경우 클래스X와 클래스 Y간의 직간접적 상, 하위 관계를 다음과 같은 요약 관계 규칙으로 정의할 수 있다.

```
class_dec_scope(X,Y):-
direct_super(X,Y).
class_dec_scope(X,Y):-
direct_super(X,Z),
class_dec_scope(Z,Y).
```

2.2 정보생성기

정보생성기는 클래스 구조와 데이터 관계에 관한 사용자 질의에 답하기 위해 클래스 정보의 개체들 사이의 관계인 요약 관계를 생성하기 위한 생성 규칙의 집합으로 구성된다. 정보생성기는 변환기에 의해 생성된 각각의 사실들(facts)에 적절한 요약 관계 생성 규칙을 적용하여 요약 관계에 대한 사용자 질의에 답하게 된다. 클래스 정보의 역공학을 위한 요약 관계를 위한 프롤로그 절들을 다음과 같이 프롤로그 규칙으로 정의하였다.

2.2.1 상하위 관계 정보

클래스 상속 계층에서 직간접적 상하위 관계를 정의한다.

```
class_dec_scope(Base, Derive):-
        direct_super(Base, Derive).
class_dec_scope(Base, Derive):-
        direct_super(Base, M),
        class_dec_scope(M, Derive).
```

2.2.2 멤버의 타입 정보

변수와 메서드로 구분되어 정의된 액세스 타입을 멤버의 액세스 타입으로 정의한다.

```
public_mem(X,Y):-
        public_memV(X,Y) ;
        public_memM(X,Y).
private_mem(X,Y):-
        private_memV(X,Y) ;
        private_memM(X,Y).
protected_mem(X,Y):-
```

```
        protected_memV(X,Y) ;
        protected_memM(X,Y).
```

2.2.3 public 상속 정보

클래스 상속 계층 내에서 public으로 유도되는 상속 관계를 정의한다.

```
public_derive(X,Y):-
        public_inherit(X,Y).
public_derive(X,Y):-
        public_inherit(X,T),
        public_derive(T,Y).
```

2.2.4 상속받은 멤버 정보

```
derived_attribute(Current,Attribute,From):-
        public_derive(From,Current),
        var_dec(From,Attribute),
        not_private_mem(From,Attribute).
derived_method(Current,Method,From):-
        public_derive(From,Current),
        method_dec(From,Method),
        not_private_mem(From,Method).
derived_mem(Current,Member,From):-
        public_derive(From,Current),
        ( var_dec(From,Member) ;
method_dec(From,Member) ),
        not_private_mem(From,Member).
```

2.2.5 클래스 정보

usable_method관계는 클래스가 사용할 수 있는 메서드를 위한 관계로서 자기 자신의 메서드이거나 상속받은 메서드이거나 다른 클래스에서 public으로 선언된 메서드인 경우에 성립한다.

```
my_mem(X,Y):-
        var_dec(X,Y) ;
        method_dec(X,Y).
usable_method(Current,Method,Where):-
        derived_method(Current, Method, Where).

usable_method(Current,Method,Where):-
        \+class_dec_scope(Where,Current),
        ( method_dec(Current, Method) ;
            \==(Current, Where),
            public_memM(Where, Method) ).
usable_method(Current,Method,Where):-
        others(Current,Where),
        public_memM(Where,Method).
others(C1,C2):-
        class(C1),
        class(C2),
        \+ ( class_dec_scope(C1,C2) ;
            class_dec_scope(C2,C1) ;
        ==(C1,C2) ).
```

2.2.6 가시성 정보

특정 클래스에서 사용 가능한 멤버 변수를 위한 관계로 자신의 멤버이거나 상속 받은 멤버이거나 public 멤버인 경우에 해당하고 private 멤버인 경우에는 이 멤버 변수가 속한 클래스 내에 이 멤버 변수를 참조하는 메서드가 존재하고, 특정 클래스에서 이 메서드를 사용할 수 있는 경우에 성립한다.

```
visible(Class, Attribute):-
        var_dec(Class, Attribute).
visible(Class, Attribute):-
        class_dec_scope(Ancestor, Class),
        derived_attribute(Class, Attribute,
        Ancestor).
visible(Class, Attribute):-
        class_dec_scope(Class, Descendent),
        var_dec(Descendent, Attribute),
        public_memV(Descendent, Attribute).
visible(Class, Attribute):-
        others(Class, Others),
        public_memV(Others, Attribute).
```

III. 시스템 평가

본 장에서는 제안된 규칙기반 역공학도구의 시스템을 평가한다. 기존의 역공학 도구들이 어떤 기준을 두고 평가된 것이 없기 때문에, 역공학 도구를 사용한 결과 프로그램 이해 정도를 갖고 측정하였다. 본 연구에서 구현한 프로그램

의 성과를 평가하기위하여 통제집단 사후 측정설계 (Posttest only control group design)를 이용한다.

3.1 평가 방법

시스템 평가는 아래와 같은 통제집단 사후 측정설계 (Posttest only control group design)를 이용하고 있다.

〈실험설계〉

실험집단 : X O_0
통제집단 : O_1

O_0 : 역공학 도구 사용집단
X : 실험처리(역공학 도구)
O_1 : 역공학 도구 미사용집단

이 실험설계는 실험대상을 무작위로 실험집단과 통제집단 등 두 집단으로 구분한다. 실험집단에게는 본 연구에서 구현한 프로그램을 통하여, 원천 프로그램의 이해를 하도록 하며, 또한 통제집단에게는 그들 스스로 원천프로그램을 이해하도록 하여, 두 집단에서 각각 이해력을 측정한다. 검증방법은 객관적 측정치에 대해서는 프로그램의 평균적 이해시간의 차이와 주관적 측정치에 대해서는 평균적 이해력간의 차이를 T-Test 분석을 한다[8].

공식 1 (두 모집단의 분산이 다를 때):

$$t=\frac{\overline{x_1}-\overline{x_2}}{\sqrt{S_1^2{}_{n_1}+S_1^2}} \qquad S_L^2=\frac{\sum(x_i-\overline{x_1})^2}{n_{1-1}}$$

공식 2 (두 모집단의 분산이 동일할 때):

$$t=\frac{\overline{x_1}-\overline{x_2}}{\sqrt{S_1^{2/}{}_{n_1}+S_1^2}} \qquad S_p^2=\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{N_{1+N_{2-2}}}$$

x_1 : 역공학 사용자 집단 평균
x_2 : 미용자 집단 평균
s_1 : 역공학 사용자 집단의 표준오차
s_2 : 미사용자 집단의 표준오차
N_1 : 역공학 사용자 집단의 수
N_2 : 미사용자 집단의 수

3.2 평가 측정치의 타당성 및 신뢰성 분석

역공학 도구의 성능평가는 객관적 측정치와 주관적 측정치를 동시에 이용하고 있다. 객관적 측정치는 역공학 도구 사용자와 미사용자들의 프로그램 이해시간을 직접 측정하였다. 그리고 주관적 측정치는 프로그램의 이해정도를 다음과 같은 10개의 항목으로 측정하였다. 즉, 문서로서의 이해 · 사용된 지역변수정보 · 사용된 광역변수정보 · 함수정보 · 사용된 데이터베이스정보 · 입력정보 · 출력정보 · 모듈간의 호출관계 · 모듈간이 데이터 흐름관계 · 모듈에서 참조되는 변수 등의 이해정도에 대해서 응답하도록 하였다.

이러한 측정치들이 이해도를 평가하는 데 타당하며, 또한 믿을 만한 측정치인가를 분석하기 위하여 요인분석(factor analysis)과 신뢰성분석(reliability analysis), 그리고 상관관계분석(correlation analysis)을 실시하였다.

요인분석은 다수의 변수들 간의 상관관계를 기초로 많은 변수들 속에 내재하는 체계적인 구조를 발견하려는 기법으로, 본 분석에서 중심이 되는 이해도는 어떠한 구조를 가지는 가를 알아보려는 것이다. 여기서 요인분석은 주성분분석(principal component analysis)을 통하여, 10개의 이해력 측정치들 간의 내재적인 구조를 추출하였다.

3.3 이해도 차이검증

역공학도구 사용자 집단과 미사용자 집단간의 이해력의 객관적 측정치와 주관적 측정치에 대한 차이를 T-test분석으로 실시하였다. 10개 항목의 주관적 측정치에 대해서는 요인분석의 결과에 의하면 하나의 차원으로 구성되어 있어, 이들 측정치를 합하여 이해도라는 하나의 평가도구로 만들어 두 집단간의 차이검증을 실시하였다. 또한, 개별 10개의 측정 각각에 대해서도 두 집단간의 차이검증을 실시하였다.

표 2. 이해도차이 분석표
Table. 2 Analysis table for Understanding

집단 / 차이	평균	표준편차	t값	확률값
사용집단	7.15	1.107	6.54	0.00
미사용집단	4.86	0.654		

10개의 측정치중 지역변수, 광역변수, 함수정보, 사용된 데이터베이스 정보, 입력정보, 모듈간의 호출관계, 모듈간의 데이터흐름 관계 등 7개 측정치가 역공학 미사용자 집단보다 사용자 집단에서 이해력이 높은 것으로 나타났다. 문서로서의 이해와 출력정보 그리고 모듈에서 참조되는 변수 등은 사용자 집단과 미사용자 집단간의 이해력의 차이는 비유

의적이나 평균은 역시 높게 나타나고 있다. 더욱이, 문서로서의 이해에 대한 두 집단간의 차이를 검증할 때, 유의성 기준을 0.10으로 완화시키면 이 측정치에 대해서도 역공학 도구 사용자 집단이 이해력이 높다고 할 수 있다.

따라서 분석결과에 의하면, 프로그램 이해시간을 나타내는 객관적 측정치와 종합항목인 이해도에서 역공학도구 사용자 집단이 미사용자 집단보다 우수하게 나타났고, 개별 항목에 대해서도 역공학 도구 사용자집단이 모두 높은 이해력정도를 보이고 있어, 이 역공학 도구의 사용이 효율적임이 입증되었다.

Ⅳ. 결 론

기존 역공학 도구들은 코드 분석에서부터 시작하여 적절한 보고서와 정보들을 생산함에도 불구하고 유지 보수 자들을 위한 중요한 정보가 충분하지 않으며, 이러한 도구들은 미리 정의된 보고서들을 작성하고, 정해진 질의들에 대해서만 답을 제공하므로 쉽게 다른 운영 환경으로 전환하기 어려우며 비유연적이다.

이러한 문제를 해결하기 위하여 본 논문에서는 프롤로그에서 제공되는 논리 프로그래밍 패러다임을 이용하여 프로그램 재구성기법을 제시하였다. 추출된 클래스 정보는 논리 프로그래밍 언어로 변환되어 정보 저장소에 저장된다. 저장된 프롤로그 사실들은 유지 보수자와 상호작용적으로 프로그램에 대한 정보를 질의응답을 통해 전달할 수 있는 새로운 패러다임을 갖고 있다. 따라서 생성된 클래스 정보로부터 역공학 정보를 얻을 수 있고, 클래스 정보로부터 프로그램을 이해하는데 유용하게 사용될 수 있도록 하였다. 또한 제안된 규칙기반 역공학도구의 시스템을 평가하였다. 본 논문에서는 구현한 프로그램의 성과를 평가하기위하여 통제집단 사후 측정설계(Posttest only control group design)를 이용하였다. 앞으로의 과제는 이러한 역공학 도구들이 완전 자동화되어지고, 평가의 기준도 표준화 가 이루어져야 할 것이다.

참고문헌

[1] Desmond F.D'Souza & Alan C.Will, "Object, Components and Frameworks with UML", Addison Wesley.1997

[2] Szyperski,C.," Component Software : Beyond Object Oriented Programming", Addison Wesley Longman, 1998

[3] Han J, "Charecterization of Components", First Int'l Workshop on Component-Based Software Engineering, in conjunction with ICSE'98, Kyoto, 1998

[4] S.Yacoub, H.Ammar, and A.Mili, "A Model for classifying Component Interface", IEEE Computer, Sep/Oct 1998

[5] K. Vaidtanathan, R .E. Harper, S. W. Hunter and K. S. Trivedi, "Analysis and Implimentation of Software Rejuvenation in Cluster Systems", ACM SIGMETRICS 2001 / Performance 2001, June 2001

[6] C. McClure, "The Three Rs of Software Automation : Reengineering, Repository, and Reusability, Prentice Hall, 1992

[7] 최은만 "A Study on Software Reuse System Using Reverse Engineering" 정보처리학회논문지, Vol.4, No.1, pp 97-109, 1997

[8] 이주복 "기초통계학" 464쪽 정익사, 2003년 2월

[9] 진영배, 왕창종, "4세대언어에서의 역공학 환경 구성," 정보처리논문지 제2권4호 pp.509-523 1995년 7월.

[10] 진영배 "프로그램 변환 및 클래스 추출기법의 설계" 한국OA학회 논문지 제3권3호 pp.64-71 1998년10월

저 자 소 개

진 영 배

1980년 인하대학교 공과대학(공학사)

1985년 인하대학교 대학원 전자계산과(이학석사)

1996년 인하대학교 대학원 전자계산전공(이학박사)

1993년 ~ 현재 충청대학 컴퓨터학부 부교수