소프트웨어 품질평가 도구(ESCORT)의 설계 및 구현

양 해 술† 권 기 현†† 이 하 용††
조 영 식†† 이 용 근†† 박 정 호††† 허 태 경††††

요 약

본 연구는 소프트웨어의 품질을 평가하기 위한 소프트웨어 품질평가 도구의 개발에 관한 것으로 소프트웨어 품질을 평가하기 위한 체계를 구축하고 소프트웨어 품질평가 활동을 지원하는 소프트웨어 품질평가 도구를 개발하는 것을 목적으로 한다. 소프트웨어 품질평가 체계는 GQM(Goal-Question-Metrics) 방법론에 의거하여 프로그램의 기능성, 유지보수성, 복잡성 및 객체지향성에 대한 부분을 포함하는 체계로 구성한다. 또한 소프트웨어 품질평가 도구는 개발자, 구매자 및 이용자간의 주관적인 요소를 배제한 형태로 자동화된 평가를 행하는 것으로 소프트웨어 분석 시스템과 분석 과정을 일관성있게 지원하는 사용자 인터페이스에 대한 부분으로 구분된다. 본 논문의 목적은 소프트웨어 품질을 소프트웨어 개발 중이나 개발 완료 후에 평가하여 소프트웨어 개발 비용과 유지보수 비용을 감소함으로써 소프트웨어의 생산성을 향상시키는 것이다.

Development of Software Quality Assessment Tool

Hae Sool Yang†, Ki Hyeon Kweon††, Ha Yong Lee††
Young Sik Cho††, Ryong Geun Rhee††, Jung Ho Park††† and Tae Kyung Huh††††

ABSTRACT

The development of automated quality evaluation tool is desperately needed to decrease the cost of maintenance, to measure quality of source program in developing course for the purpose of developing high quality software. Main goal in this paper is to develop the automated tool for software quality evaluation related with high quality and high reliability of software. In this paper, we proposed the four software quality model to evaluate software quality; functional-size model, understandability model, complexity model and object-oriented model. Also, we made a scanner and a parser to analyze the c and c++ source program and to produce the metrics and function value. The measurement value is statistically analyzed for the distribution of the measurement value. we can extracted the characteristics of measurement value and this one is assisted to make scores of software quality evaluation. Finally, we made a software quality evaluation tool to support software evaluation activities.

1. 서 론

소프트웨어 품질을 향상시키기 위한 방법으로 소프트웨어 품질 요인에 대한 정량적이고, 품질 측정 결과에 대한 가시적인 품질관리 방법이 요구되고 있다[3, 6, 11]. 이것은 개발자, 구매자, 사용자가 소프트웨어 품질에 대한 신뢰성과 성능에 대한 의사 결정(decision making)을 하는데 하나의 지표가 될 수 있기 때문이다. 소프트웨어의 신뢰성과 성능 평가는 소프트웨어를 사용하고 테스트하는 등의 실제적인 방법에 의해 이루어질 수 있으나 개발자, 구매자, 사용자간의 이해 및 요구 정도의 차이에 의해 주관적인 요소가 포함되어 객관적인 평가를 하는데 매우 많은 어려움이 있으므로 소프트웨어 공학의 공리(axiom)에 의거한 정량적이고 가시적인 품질관리 방법이 필요하다.

· 본 연구는 한국통신 장기기초 연구사업(협약서 : 94-1)의 연구비 지원의 일부로 이루어졌음.
† 종신회원 : 강원대학교 전자계산학과 교수
†† 종신회원 : 강원대학교 전자계산학과 박사과정
††† 종신회원 : 선문대학교 전자계산학과 조교수
†††† 정 회 원 : 한국통신 품질보증단 개발품질 부장
논문접수 : 1994년 10월 1일, 심사완료 : 1994년 12월 1일

고품질이고 보수가 용이한 소프트웨어를 개발하기 위해서는 소프트웨어 생명주기(life cycle) 각 단계에서 중간 제품을 분석하고, 소프트웨어 비용과 품질에 영향을 주는 요인을 측정할 수 있는 정량적인 방법과 이런 과정을 효율적으로 지원할 수 있는 품질관리 도구가 요구된다[3].

본 논문에서는 소프트웨어의 품질 요인항목을 정량적으로 관리하여 소프트웨어의 품질을 평가하는 자동화 도구(automated tool)를 구현하였다.

이를 위해서 절차적인 언어인 C 언어와 객체지향 언어인 C++ 언어를 대상으로 하여 소프트웨어 품질을 정량적으로 관리하기 위해 GQM (Goal-Question-Metrics)[9] 방식에 의해 기능 사이즈 모델, 이해 용이성 모델, 복잡성 모델과 객체지향 모델을 제안하고 각 모델을 모델의 특성에 따라 적합한 프로덕트 메트릭스(product metrics)[6]로 구성하여 체계를 마련하였다. 그리고 정적분석(static analysis)[17] 방법을 이용하여 각 모델별 메트릭스에 대하여 기본적인 사항(elementary measurement)을 측정하고 분류 및 저장시키는 파서(parser)를 구현하였으며 파서의 출력으로 얻어진 분석 화일을 이용하여 소프트웨어의 측정치(estimation value) 및 메트릭스 값을 추출하는 분석 시스템을 구현하였고, 분석 시스템으로부터 추출된 값의 통계적 분석을 통해 측정값을 득점값(score value)으로 변환하여 테스트 단위, 메트릭스 단위 및 모델 단위로 종합된 평가(assessment)값을 도출하였다. 또한, 도출된 소프트웨어 품질측정에 대한 상황을 명확하고 직관적으로 나타내기 위해 X-Window를 이용하여 소프트웨어 품질평가 상황을 가시화하였다.

본 논문의 구성 및 전개는 2장에서 소프트웨어 품질관리에 대한 관련 연구를 살펴보고, 3장에서는 본 논문에서 제안한 품질관리 도구의 품질평가 체계에 대하여 알아보며, 4장에서는 제안한 모델에 대하여 품질측정 요소를 정량적으로 측정하는 내부 분석 시스템의 구현 사항에 대하여 설명하였다. 그리고 5장에서는 분석기를 통해 추출된 값에 대하여 값의 비교를 위한 통계적 분석 및 변환 방법에 대하여 설명하였으며, 6장에서는 4장에서 구현한 분석기에 5장의 척도 변환의 방법을 도입하여 득점값을 추출하고 평가 결과를 명확하게 출력하는 출력 형태를 제시하였다. 마지막으로 7장에서 본 연구의 결과와 향후 연구 과제를 제시하였다.

2. 관련 연구

2.1 소프트웨어 품질관리

2.1.1 품질관리의 개요

소프트웨어 공학(software engineering)이란 소프트웨어에 대하여 좋은 품질의 소프트웨어 시스템이 높은 생산성으로 개발될 수 있도록 방법과 도구를 개발하는 것으로 Pressman[17]은 실제 기계에 대하여 효과적이고 신뢰성있게 동작할 수 있는 소프트웨어를 적은 비용으로 얻기 위한 공학적 원리의 사용 또는 설립 과정이라고 정의하고 있다.

품질관리의 목표는 소프트웨어의 품질을 테스트하고, 보수성있는 모듈과 부품 작성을 위한 체계를 정립하고, 일관성있는 품질관리 환경을 제공함으로써 생명주기의 유지보수 비용과 노력을 감소시키는 것이다.

2.1.2 절차적인 언어의 품질관리

1) 프로그램 크기에 의한 복잡도 산출

프로그램의 외형적 정보 즉, 문장(statement), 오퍼레이터(operator), 오퍼랜드(operand) 등으로 프로그램의 수량적 복잡도를 측정하는 방법이다.

① 스텝 수에 의한 방법

프로그램의 문장 수에 의한 추정 방법으로 프로그램 언어에 제한을 받지 않고 측정 방법이 단순하며 외형적인 프로그램 작성 노력을 측정할 수 있다[5].

라인당 문장의 수와는 상관없이 공백(blank)과 설명문(comment)을 제외한 모든 라인의 수, 특히 프로그램 머리부(header), 선언 부분, 비실행문을 모두 포함한다.

② Halstead의 소프트웨어 과학(software science) 방법

Halstead는 프로그램 규모가 프로그래밍 언어별로 크게 차이가 나는 것을 발견하고는 라인당

가중치를 고려해 줄 수 있는 토큰(token)의 개수를 가지고 소프트웨어의 규모를 측정하였다[10].

2) McC abe의 사이클로매틱 복잡도 산출

서로 독립적인 경로(path)의 수를 계산함으로써 프로그램 제어흐름의 논리적인 복잡도를 측정하는 메트릭이다[14].

3) 기능점 분석(Functional Point Analysis)

Albrecht[1]에 의해 처음 제안된 규모 측정 방법으로 LOC 대신 소프트웨어가 수행하는 기능(function)의 양을 사용자의 관점에서 측정하는 방법이다.

4) 데이타 양 메트릭스(Data Amount Metrics)

프로그램 내에서 입력, 출력 및 처리되는 데이타의 양을 측정하는 메트릭스이다. 데이타의 양은 전후 참조 리스트(cross-reference list)의 엔티티(entity)들인 키워드(keyword)로 사용되는 프로그램명 및 변수명 그리고 함수명 등의 수를 셈으로써 측정된다[5].

5) 정보 흐름 메트릭스(Information Flow Metrics)

정보 흐름 메트릭스란 프로그램 모듈(module) 간의 데이타 흐름을 다루는 메트릭스로 전체 시스템의 이해에 목적을 둔다. Henry와 Kafura [18]에 의해 제시된 정보 흐름 복잡도 메트릭은 부프로그램의 fan-in과 fan-out의 수를 기본으로 하여 측정된다.

2.1.3 객체지향 언어의 품질관리

객체지향 분석 및 설계 등의 방법이 급속히 보급됨에 따라 연구자들은 품질 좋은 개발 방법론의 필요성을 느끼게 되었다. 따라서 구조적 분석·설계에서 모듈의 응집도(cohesion)와 모듈간의 결합도(coupling) 등과 같은 이론들이 객체지향 설계에서도 요청되어 Moreau와 Dominick[16, 17], Chidamber와 Kemerer[4], Jenson과 Bartley[13], Dumke[8] 그리고 Jacobson[12] 등이 객체지향 프로그램의 복잡도 산출과 품질 측정을 위한 연구를 하였다.

3. 소프트웨어 품질관리 체계

3.1 품질관리 목표

3.1.1 품질관리 모델

임의의 사물을 측정한다고 할 때, 보통 이미 존재하는 몇몇 엔티티(entity)에 대하여 평가(assessment)하는 것을 의미한다. 그러나 때로는 많은 상황에서 아직 존재하지 않는 몇몇 개체의 속성(attribute)을 예측하기를 원하는 경우도 있다. 예를 들어, 한번 사용된 적이 있는 소프트웨어 시스템의 신뢰성을 적절히 평가(assessment)할 수 있겠지만, 미 개발 중인 시스템이라도 축적된 지식에 근거하여 유사한 형태의 신뢰성에 대하여 측정할 수 있을 것이다.

이러한 측정(measurement)과 예측(prediction) 사이에서 Fenton은 모델(model)을 '객체(object)의 추상적 표현'[9]이라고 정의하고 있다. 이 정의는 매우 일반적인 것으로서 모델에는 여러 가지 유형이 있다는 것을 암시하고 있다.

모델은 소프트웨어 측정에 관련하여 필수적으로 두 가지 부류로 분류된다.

① 모델은 다양한 제품(product), 절차(process), 자원(resources)에 대한 추상적인 표현이다. 모델은 측정치를 모호함없이 정의하기 위해서 요구되어진다. 명확하고 잘 정의된 측정 방법인 LOC로 측정치를 정의한다고 해도 LOC에 대한 형식적인 모델이 요구된다.

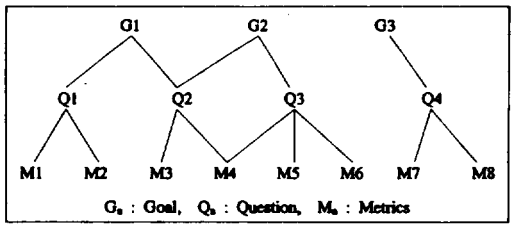
② 모델은 엔티티(entity)의 속성 사이의 관계(relationship)에 대한 추상적 표현이다. 이들 모델은 보통은 수학적인 공식으로 형성된 2가지 이상의 측정식 또는 메트릭스로 구성되어진다.

모델은 속성간의 관련(association)을 정의하는 것이다. 이러한 모델을 이용하여 유용한 예측 시스템(prediction system)을 형성하는 것이 가능하게 된다.

3.1.2 메트릭스 결정

소프트웨어 품질관리를 위한 메트릭스를 결정

하는 방법은 하향식 방법(top-down method)과 상향식 방법(bottom-up method)이 있다[9]. 하향식 방법은 먼저 프로젝트 관리의 목표를 세우고 이 목표를 달성하기 위해 필요한 메트릭스를 결정하는 방법이다. 상향식 방법은 품질관리에 어떠한 엔티티(entity)들이 있고 이 엔티티들에는 어떤 속성(attribute)이 있는가를 살핀 후 이중에서 필요한 메트릭스를 결정하는 방법이다. 이 두 방법은 상호 보완적으로 적절히 동시에 사용될 수 있다.



(그림 1) GQM 방식의 예
(Fig. 1) Example of GQM Approach

본 논문에서는 하향식 방법의 하나인 GQM (Goal-Question-Metrics)[9] 방법을 적용하여 품질 목표를 정하고 품질 목표와 관련된 메트릭스를 결정하였다(그림 1).

3.2 소프트웨어 품질관리 모델과 요인항목

소프트웨어 품질평가 자동화 도구 ESCORT(Evaluation System of C/C++ ORiented qualiTy)는 지금까지의 연구를 통하여 외부 특성과 내부 특성을 기초로 하여 실제로 C와 C++ 프로그램을 분석하여 평가하기 위한 소프트웨어 품질특성의 요인항목을 추출하였다. 이 품질특성을 정량적으로 측정하기 위해 기능 사이즈 모델, 이해 용이성 모델, 복잡성 모델 및 객체지향 모델을 구성하였다. 각 항목별로 구현 시에 요구되는 사항을 살펴보면 다음과 같다.

3.2.1 기능 사이즈 모델

기능 사이즈 모델은 모듈 하나를 유지하는 기능의 수를 측정하여 정량화한 것이다. 소스 프로그램의 각각의 모듈이 유지하고 있는 기능이 작고, 하나의 모듈당 한 개의 기능을 가지고 있는 것이 바람직하다는 점을 고려한 모델이다. 모듈당 기능이 작으면 보수할 때 다른 모듈에 영향을 적게 주며, 기능 추출이 간단하다.

① 출구수
② 블럭수
③ 명령어 카테고리 수
④ 함수군의 수
⑤ 최외 블럭 수
⑥ 두번째 블럭 수

3.2.2 이해 용이성 모델

이해 용이성 모델은 소스 프로그램의 읽기 쉬움을 정량화한 것으로 작성된 소스 프로그램이 시각적으로 읽기 쉽다면 프로그램을 빠르게 이해할 수 있다는 생각을 기초로 한다. 즉, 프로그램을 쉽게 이해할 수 있다면 보수 시에 영향을 주는 범위를 명확하게 알 수 있다.

① 주석율
② 주석 위치
③ 함수의 주석수
④ 변수의 유효범위 평균치
⑤ 스텝 수
⑥ 출구 수
⑦ 블럭 평균 길이
⑧ 내부 변수의 밀도
⑨ 외부 변수의 밀도
⑩ GOTO문 수

3.2.3 복잡성 모델

복잡성 모델은 소스 프로그램의 논리의 복잡성을 정량화한 모델이며, 소스 프로그램의 논리가 단순 명쾌한 것이 바람직하다는 생각을 기초로 한다.

다음의 7가지 메트릭스를 이용하여 구성한 모델이다.

① 실행문 수
② Halstead의 어휘 사이즈
③ Halstead의 프로그램 길이
④ Halstead의 프로그램 볼륨
⑤ Halstead의 프로그램 수준
⑥ Halstead의 프로그램 난이도 및 노력도
⑦ McCabe의 척도

3.2.4 객체지향 모델

객체지향 모델에서는 클래스의 문자수와 클래스의 라인수를 포함하여 Kemerer와 Chidamber가 객체지향의 설계 규모와 복잡도에 영향을 주는 요소를 측정하기 위해 측정 이론(measurement theory)과 숙련된 객체지향 소프트웨어 개발자의

통찰력을 사용하여 제안한 6가지 척도를 사용한다[4].

① 클래스당 문자수
② 클래스당 라인수
③ 클래스당 가중치를 가진 메소드 (WMC, Weighted Methods per Class)
④ 상속 트리의 깊이(DIT, Depth of Inheritance Tree)
⑤ 서브 클래스의 수(NOC, Number of Children)
⑥ 객체간의 결합도(CBO, Coupling Between Objects)
⑦ 클라이언트 클래스에 대한 메소드 호출 정도(RFC, Response For a Class)
⑧ 메소드에서 응집도의 결함 정도 (LCOM, Lack of Cohesion in Methods)

4. 소프트웨어 품질관리 내부 시스템의 구현

4.1 분석기의 기능 및 제한 사항

소프트웨어 품질에 영향을 미치는 요인에 대한 정량적인 품질관리를 위하여 정적분석(static analysis)[17]의 방법을 이용하여 각 모델별 메트릭스에 대하여 기본적인 측정 사항(elementary measurement)을 측정하고 분류 및 저장시키는 파서(parser)를 구현하였으며 파서의 출력으로 얻어진 분석 화일을 이용하여 소프트웨어의 측정치(estimation value) 및 메트릭스(metrics) 값을 추출하는 분석 시스템을 구현하였다.

분석기는 품질관리 도구의 내부 시스템으로 품질관리 도구의 분석기의 기능을 수행하며, 다음과 같은 기능 및 제한 사항이 있다.

- 대상으로 하는 언어는 절차적인 언어인 C 언어와 객체지향 언어인 C++ 언어로 한다.
- 입력되는 프로그램은 구문 에러(syntax error)가 제거된 프로그램으로 한다.
- 프로그램의 입력은 모듈 단위로 한다.
- 기능 사이즈, 이해 용이성, 복잡성 모델은 메소드를 포함한 함수를 테스트 단위로 한다.
- 객체지향 모델은 클래스를 테스트 단위로 한다.
- 모델별 테스트 단위에 대한 측정 결과는 계산값 리스트로 출력된다.
- 분석기는 UNIX 운영체제에서 동작한다.

4.2 품질관리 내부 시스템의 구조 및 흐름

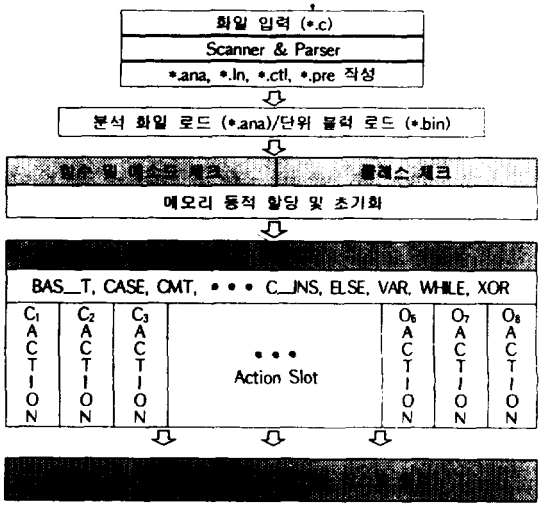
내부 시스템은 컴파일 가능한 C 언어 및 C++ 프로그램을 입력으로 하여 스캐너(scanner)와 파서(parser)를 이용해 분석 화일을 생성하고 요구되는 사항을 추출하며, 필요한 모듈은 적절한 위치에 화일로 저장시키는 기능을 담당한다. 이러한 산출물은 드라이버에 의해서 처리되어 본 도구에서 제안한 모델에 대한 기본적인 측정 값을 도출하고 메트릭스를 계산해서 계산값 리스트를 저장 및 출력하게 된다. 계산값 리스트는 이후의 품질평가 단계에서 중요한 기본 자료로 사용되어 품질평가 활동을 지원하게 된다. 이러한 내부 시스템의 구성을 (그림 2)에 나타내었다.

4.2.1 입력 단계

입력 단계는 소프트웨어 품질관리 도구에서 대상으로 하는 언어인 C 언어와 C++ 언어를 입력하는 단계이다. 이 단계를 통해 구성 화일에는 입력되는 소스 프로그램의 위치와 화일명들이 기록된다.

4.2.2 프리프로세싱 단계

프리프로세싱 단계는 스캐닝, 파싱 및 처리 루



(그림 2) 품질관리 내부 시스템의 흐름도
(Fig. 2) Flow of analysis system

틴에서 사용될 입력 화일을 작성하기 위한 기본적인 전처리를 수행하는 단계이다.

이 단계에서 처리할 사항은 다음과 같다.

- ○ 입력되는 프로그램 모듈명 결정
- ○ 파서에 입력이 되는 소스를 위해 불필요한 부분 삭제
- ○ 입력되는 화일에 대하여 모듈별 단어 수, 실행문 수, 라인수 계산
- ○ 주석의 총 문자수 및 주석 정보 테이블 작성
- ○ 블럭 구분 화일 작성

4.2.3 렉시칼 분석(Lexical Analysis) 단계

주어진 입력화일(*.pre)을 토큰으로 분류하여 파서에게 전달하는 단계이다. 이때 가장 기본적인 측정치는 직접 구해 *.ana에 저장한다.

4.2.4 구문 분석(Syntactic Analysis) 단계

파서 분석 단계는 렉시칼 분석 단계로부터 전달되는 토큰을 파서 분석을 이용하여 프로그램 흐름에 따른 한 차원 높은 단계의 정보를 구하여 *.ana 화일에 가능한 모든 정보를 수록한다.

이 단계에서 처리할 사항은 다음과 같다.

- ○ 변수, 함수, if, else, for, switch, do, while, class 등의 자료 처리
- ○ 멤버 변수, 오퍼레이터, 오퍼랜드, 내장 기본 형(type), 인스턴스의 자료 처리
- ○ 기본 함수, 구조체, 열거형의 선언, 전역 변수, 함수 및 멤버 함수의 파라메타의 자료 처리
- ○ 사용된 클래스의 멤버 함수 등의 기본 자료 처리
- ○ 기타 파서에 의해서 추출되는 자료 처리
- ○ extern으로 선언된 변수 및 함수에 대한 정보처리

함수, 클래스, if, switch, while, for 등의 블럭에 대한 자료는 블럭별로 별도의 화일에 블럭을 구성하는 기본 위치, 사이즈 등을 계산하여 지정한 디렉토리에 저장한다.

4.2.5 처리 및 가공 단계(Action and Operation)

파싱의 전 단계까지 걸쳐 추출된 모든 자료를 이용하여 모델별, 요인항목별, 메트릭스별로 구분하여 값을 도출하는 단계이다.

4.2.6 출력 단계

출력 단계는 처리 및 가공을 거쳐서 구해진 하나의 메트릭스에 대한 측정치를 저장 및 출력하는 단계이다.

각 단계별로 생성되고 사용된 화일 흐름을 〈표 1〉에 나타낸다.

〈표 1〉 품질관리 내부 시스템 화일 처리 과정
〈Table 1〉 File processing step of analysis system

*.ana / *.ctl	입 력	C++ 소스 프로그램 *.ctl
*.ln / *.pre	전처리	*.ctl / *.ana / *.ctl
	렉시칼 분석	*.pre / *.ana
	구문 분석	*.pre / *.ln / *.ana
	동작 및 가공 처리	*.ln / *.ana 블럭 별로 저장된 화일
*.rst	최종 측정	*.ctl
	출 력	

4.3 출력 결과

소프트웨어 품질관리 내부 시스템의 각 단계를 통해 생성되는 화일과 자료들을 이용해서 도출한 메트릭스 값의 제시 형태는 다음과 같다.

4.3.1 중간 생성물

1) 블럭 및 라인 정보 화일

C 언어 및 C++ 언어는 '{'와'}'로 블럭이 구

```
554@   0@class Action
555@   2@{
557@   0@private:
559@   0@   int timeatstart;
561@   0@   int timeremaining;
563@   0@public:
565@   0@   Action();
567@   0@   int continues(void);
569@   0@   void settime(int secs);
571@   0@   int gettime(void);
573@   0@   void reducetime(int secs);
575@   0@   void perform(void);
577@   0@   void display(void);
579@   0@   void results(void);
582@   1@};
```

(그림 3) 블럭 및 라인 정보 화일
(Fig. 3) Block and line file

분되고 있으므로 '{'와 '}'에 대한 위치 정보를 정확히 유지하는 것이 필요하다. 블럭화일에는 '{'와 '}'에 대한 정보를 계산하여 *.ln 화일에 (그림 3)과 같은 형태로 소스루틴과 함께 수록한다.

2) 파서 입력 화일

파서 입력 화일은 블럭 레코드 화일과 소스 프로그램의 위치를 같게 하기 위해서 블럭 레코드 화일에서부터 작성되어진다. 블럭 레코드 화일에서 파서 입력 화일로 전환하기 위해서 간단한 Lex 문법을 사용하여 처리한다.

생성된 *.pre 화일에서는 *.ln 화일에서 #include 등의 전처리 문과 주석문을 제거하여 파서의 입력으로 사용하게 된다. 이때 본래의 프로그램의 위치는 변하지 않는다.

3) 블럭 화일

모듈의 주성분인 클래스, 매소드, 함수 및 do 문, if 문 등은 별도로 저장되는 것이 필요한데 이러한 동작을 파서에서 수행하게 된다. 블럭을 저장하고 있는 화일이 저장하고 있는 내용에는 모듈의 식별자, 모듈의 범위 그리고 모듈의 총 문자수, 모듈의 세미콜론 수 및 모듈의 내용 등이 있다. 이러한 블럭 화일의 예를 (그림 4)에 나타내었다.

```
FILE=MODULE/CF1.bin
39 42 38 1
virtual void display()

{
   cout<< hour<< ':' << minute<< ':' << second;
}
```

(그림 4) 블럭 화일
(Fig. 4) Block file

4) 분석화일

분석 화일은 파싱 과정을 통해 가장 중요하고 많은 양의 자료가 저장되는 곳으로 변수 하나의 타입과 유효 범위에서 함수의 위치 및 구조적인 데이타에 이르기까지 많은 자료가 저장된다.

화일을 입력하여 내부 시스템을 통해 처리한 후 분석 화일에 저장되는 분석 화일의 예를 (그림 5)에 제시한다. 분석 화일에는 데이타명, 데이타의 타입, 데이타가 위치한 라인 번호 또는 데이타의 개수 등과 같은 기본 측정치(elementary measurement)가 누적되어 있는 형태를 취한다.

```
##############################
#        ANALYSIS FILE       #
##############################
AND@  168@   1
ASSIGN@  187@  12
CMT@ 40@ 7

"중간 생략"

P_BAS@   34@hr@int
VAR@  294@var_4
V_MF@   39@display()@void
WHILE@  299@   4
XOR@  253@   2
END
```

(그림 5) 분석 화일의 부분 예
(Fig. 5) A part of analysis file

4.3.2 모델의 계산값 리스트

모델의 계산값 리스트는 모델별 메트릭스에 대한 값을 클래스의 매소드와 함수의 단위에서 측정값을 구한 리스트이다.

ID	1) ST	2) RA	3) SZ	4) LT	5) VM	6) LV	7) DS	8) MC
MODULE/FN1.bin	16.00	0.06	17.00	45.00	184.00	0.10	1839.00	3.00
MODULE/FN2.bin	9.00	0.19	5.00	5.00	12.00	0.44	26.00	1.00
MODULE/FN3.bin	18.00	0.15	13.00	22.00	81.00	0.35	235.00	1.00
MODULE/FN4.bin	12.00	0.11	13.00	25.00	93.00	0.19	476.00	1.00
				‖				
MODULE/CF3.bin	2.00	0.00	8.00	7.00	21.00	0.40	52.00	1.00
MODULE/CF4.bin	2.00	0.00	7.00	8.00	22.00	0.53	42.00	1.00
MODULE/CF5.bin	2.00	0.00	7.00	8.00	22.00	0.53	42.00	1.00
MODULE/CF6.bin	2.00	0.00	8.00	7.00	21.00	0.40	52.00	1.00
MODULE/CF7.bin	2.00	0.00	8.00	7.00	21.00	0.40	52.00	1.00
VARIATION	25.08	0.13	18.18	43.14	201.49	0.22	2906.73	3.10
STANDARD Dev.	33.94	0.08	10.01	48.07	267.42	0.15	7295.10	3.57

(그림 6) 복잡성 모델의 계산값 리스트
(Fig. 6) Computational list of complexity model

5. 메트릭스 계산값의 분석

5.1 품질 측정값의 척도(scale)

5.1.1 품질 측정치의 변환 방법

계산값 리스트의 값을 살펴보면 메트릭스와 테스트되는 단위 함수에 대한 값의 범위가 주석율의 경우에는 1에서 0의 사이의 값으로 계산값 리스트가 나타나고, 문자수를 비롯한 클래스의 응집도와 결합도의 경우에는 0에서 부터 n까지의 값으로 나타남을 알 수 있다. 이러한 다양한 범위의 측정치는 서로 값의 대소를 비교하는데 어

려움이 따르게 되어 측정치의 결과를 가시화 하는데 어려움이 있다. 따라서 이러한 측정치를 일정한 형태의 척도로 변환할 필요성이 있다.

측정치를 일정한 형태로 변환하는 방법에는 여러방법이 있으나 Fenton의 연구에 따르면 〈표 2〉와 같은 형태로 변환시킬 수 있다[9].

〈표 2〉 측정치의 척도
〈Table 2〉 Scale of measurement

Nominal	M' = F(M) (F 1-1 mapping)	엔티티들의 레벨화 또는 분류
Ordinal	M' = F(M) (F monotonic increasing ie M(x) ≥ M(y) ⇒ M'(x) ≥ M'(y))	우선권, 경도, 공기의 질 지능 테스트(원시 득점)
Interval	M' = αM + β (α > 0)	시간(달력) 온도(화씨, 섭씨) 지능 시험(*표준 득점)
Ratio	M' = αM (α > 0)	시간 간격, 길이, 온도(절대온도)
Absolute	M' = M	계수(Counting)

첫째, Nominal 척도는 원시 값과 변환값 사이에 1:1 맵핑이 가능한 경우에 적용할 수 있고 둘째, Ordinal 척도는 단조 증가함수 집합의 경우에 사용되며, 셋째, Interval 척도는 단조 증가함수에 의존해서 원시 값을 변환시킬 수 없는 경우이며 값의 변환 형태가 임의의 간격을 두고 있는 경우에 사용하게 된다. 넷째, Ratio 척도는 엔티티의 속성(attribute)이 없는 경우를 포함해서 값이 0이 되는 경우를 포함하는 경우로서 계산값에 임의의 상수를 곱해서 비율 변환치로 변환이 요구될 때 사용되고 마지막으로 Absolute 척도는 엔티티들의 단순한 계수로서 사용되는 경우에 적용될 수 있다.

5.1.2 메트릭스 값의 유형

메트릭스는 소프트웨어에서 측정 가능한 요인들을 추출해 내는 지표[19]로서 메트릭스 값은 소프트웨어의 복잡도를 의미하게 된다. 메트릭스의 복잡도 중에서 주석율은 측정값이 높을수록 양질의 프로그램으로 생각할 수 있고, 프로그램의 사이즈를 비롯한 대부분의 메트릭스는 측정값이 낮을 수록 양질의 프로그램으로 생각할 수 있다. 따라서 본 소프트웨어 품질관리 내부 시스템에서 측정한 값을 득점(score)로 변환시키는데는 메트릭스의 특성에 따라 득점의 형태를 결정해 주어야 한다. 〈표 3〉에는 모델별로 메트릭스의 유형을 측정값이 높을수록 득점값이 높아지는 positive의 형과 측정값이 낮을 수록 득점값이 높아지는 negative의 형으로 분류한 형태를 제시하고 있다.

〈표 3〉 메트릭스의 유형
〈Table 3〉 Types of metrics

기능지향모델	출구수	Negative
	블럭수	Negative
	카테고리수	Negative
	함수군의 수	Negative
	최외블럭수	Negative
	두번째블럭수	Negative
어휘응어성	주석율	Positive
	주석위치	Positive
	함수의 주석수	Positive
	변수의유효범위 평균치	Negative
	스텝수	Negative
	출구수	Negative
	블럭평균길이	Negative
	내부변수의 밀도	Positive
	외부변수의 밀도	Positive
	GOTO문의 수	Negative
복잡성모델	Statement	Negative
	Halstead Size	Negative
	Halstead Length	Negative
	Halstead Volume	Negative
	Halstead Level	Negative
	Halstead Efforts	Negative
	McCabe V(G)	Negative
객체지향모델	Character	Negative
	Statement	Negative
	WMC	Negative
	DIT	Negative
	NOC	Negative
	CBO	Negative
	RFC	Positive
	LCOM	Negative

5.2 메트릭스 계산값의 분석

5.2.1 메트릭스 계산값의 분포

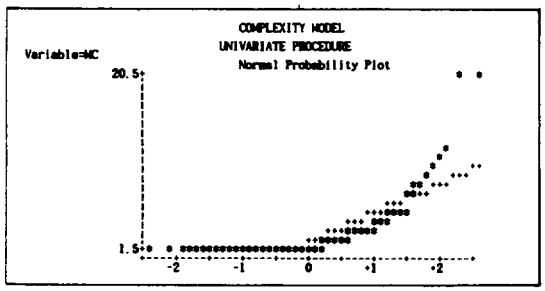
소프트웨어 품질관리 내부 시스템에 의해 측정된 자료의 분포와 성향을 파악하기 위해서 측정치의 분포도를 SAS 6.01 통계 프로그램을 이용하여 측정한 분포도를 살펴본다. 측정치의 분포 분석에 사용된 통계 프로그램의 일부를 (그림 7)에 제시한다.

복잡성 모델의 메트릭스 중 McCabe의 사이클로매틱수에 대한 분포도를 살펴보면 (그림 8)과 같은 형태로 나타나고 도수 분포를 히스토그램으로 나타내면 (그림 9)의 형태가 된다.

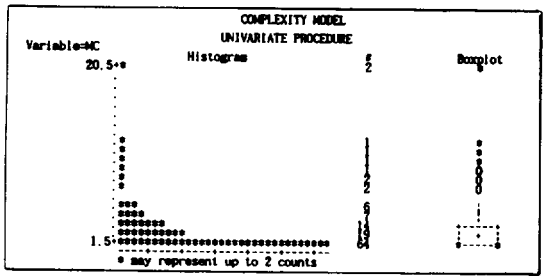
(그림 8)에서 '*'는 측정치의 분포를 나타내고, '+'는 평균과 분산에 대한 기준 참조 값으로 측정치의 분포가 정규 분포를 따른다면 기준선과 측정치의 분포가 같은 직선으로 구성되어 있어야

```
DATA ESCORT;
TITLE "COMPLEXITY MODEL";
INPUT ID $1-3 ST RA SZ LT VV LV DS MC;
LIST;
CARDS;
  1   11.00    0.56   14.00   32.00   122.00   0.24   508.00   4.00
  2   13.00    0.20   14.00   21.00    80.00   0.20   400.00   2.00
  3   25.00    0.08   23.00   60.00   271.00   0.15  1809.00   1.00
                          "중간 생략"
117    2.00    0.00    7.00    8.00    22.00   0.53    42.00   1.00
118    2.00    0.00    7.00    8.00    22.00   0.53    42.00   1.00
119    2.00    0.00    8.00    7.00    21.00   0.40    52.00   1.00
120    2.00    0.00    8.00    7.00    21.00   0.40    52.00   1.00
;

PROC CORR;
PROC UNIVARIATE PLOT;
   VAR ST;
   VAR RA;
   VAR SZ;
   VAR LT;
   VAR VV;
   VAR LV;
   VAR DS;
   VAR MC;

RUN;
```

(그림 7) SAS 통계 프로그램(부분)
(Fig. 7) Program used in Statistical Analysis



(그림 8) McCabe 척도의 분포도
(Fig. 8) Distribution of McCabe metrics



(그림 9) McCabe 척도의 히스토그램
(Fig. 9) Histogram of McCabe metrics

하고 분포 형태가 X축과 Y축에 대해서 사선의 형태를 취하게 된다. McCabe 사이클로매틱수에 대한 분포 형태는 측정치가 정규 분포에 크게 미치지 못한다는 것과 측정치에 대한 편차가 몇몇 특정 측정치에 의하여 편차의 폭을 크게 하고 있다는 것을 알 수 있다.

(그림 9)는 측정치의 도수를 히스토그램과 Box-Plot으로 나타낸 것으로 그래프 형태가 가운데를 중심으로 분포하고 있다면 정규 분포에 가깝다고 생각할 수 있으나 측정치들의 도수가 한쪽 방향으로 크게 치우쳐 있고, Box-Plot의 기준선이 하단으로 치우쳐 있음을 알 수 있다.

5.2.2 메트릭스 기준치 결정

측정된 계산값의 대소를 비교하기 위해 측정된 계산값을 서로 비교 가능한 척도인 비율(ratio) 척도로 변환하기 위해서는 각각의 모델에 대하여 메트릭스 단위로 변환 상수 α를 결정하는 것이 요구된다[9]. 본 논문에서는 변환 상수 α를 결정하기 위해서 4개의 소프트웨어에서 120개의 함수 및 매소드와 19개의 클래스를 추출하여 소프트웨어 품질관리 내부시스템에 의해 분석한 결과 모델별 메트릭스에 대해 최대값, 최소값, 중위수, 최빈수, 평균값을 〈표 4〉와 같이 추출하였다.

최대값과 최소값의 결정은 메트릭스에 대한 계산값의 구간을 설정하는 것으로 메트릭스의 구간

〈표 4〉 메트릭스의 구간
〈Table 4〉 Scope of metrics

기초사이즈	출구수	0	0.5	1	0.3	0.2
	블럭수	1	9	18	5	3
	함수군의 수	1	20	35	10	8
	최외블럭수	0	2	4	2	2
	두번째블럭수	0	3	6	2	1
이해용이성	주석율	0	0.5	1	0.3	0.2
	주석위치	0	0.5	1	0.4	0.3
	함수의 주석수	0	15	26	10	7
	변수의유효범위 평균치	0	500	1789	300	200
	스텝수	0	20	265	15	10
	출구수	2	35	208	30	15
	블럭평균길이	0	2	6	2	1
	내부변수의 밀도	0	0.5	1	0.3	0.2
	외부변수의 밀도	0	0.5	1	0.2	0.1
	GOTO문의 수	0	0	2	0	0
복잡성	Statement	2	19.075	208	12	2
	Comment Ratio	0	0.1	1	0.08	0.1
	Halstead Size	5	17.1916	60	14	8
	Halstead Length	5	39.73	238	25	9
	Halstead Volume	12	184.283	1336	98	50
	Halstead Level	0.03	0.26208	0.58	0.22	0.15
	Halstead Efforts	26	2698	46079	195.5	65
	McCabe V(G)	1	2.61667	10	1.01	1
재사용성	Character	220	650.684	2535	364	300
	Statement	89	470.11	2175	263	100
	WMC	107	727.53	3287	524	200
	DIT	1	1.684	7	1.01	1
	NOC	0	0.47368	3	0.01	0
	CBO	7	15.95	56	10	8
	RFC	2	5	10	4	4
	LCOM1	0	3	10	3	0

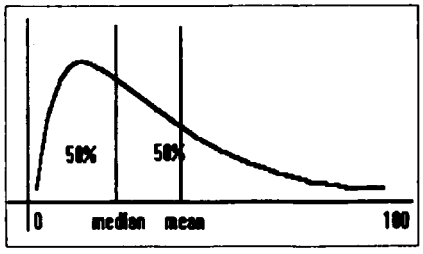
과 계산값의 분포 형태에 근거하여 계산값를 득점화할 수 있으며, 메트릭스별 득점은 소프트웨어의 여러 측면에서 소프트웨어의 품질에 대한 상황을 제시해 볼 수 있는 기본적인 토대가 된다고 할 수 있다.

5.2.3 계산값의 득점 형식

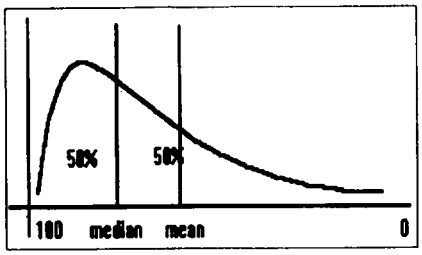
측정된 계산값를 비교 평가하기 위해서 앞절에서 기술한 품질 측정치의 변환방법 중 비율(ratio) 척도로 변환하는 방법에 대해 기술한다.

앞 절의 메트릭스의 유형과 측정값의 분석에서 기술한 바와 같이 메트릭스의 계산값는 값의 유형에 따라 negative와 positive로 분류하였고 측정값의 분석을 통해 값의 분포가 정규성을 따르지 않고 한 방향으로 강하게 치우쳐 있으므로 평균(mean)값을 이용해 득점을 구하기보다는 중위수(median)를 이용하여 득점을 구한다.

득점은 중위수를 기준으로 positive 메트릭스의 상한값과 하한값, negative 메트릭스의 상한값과 하한값으로 구분하여 득점을 구한다(그림 10).



(a) Positive 메트릭스



(b) Negative 메트릭스

(그림 10) 메트릭스 값 분포 형태
(Fig. 10) Distributed types of metrics value

1) Positive 메트릭스의 상한값

Positive 메트릭스의 측정치 중 중위수보다 큰 측정치에 대해서는 (식 1)에 의해 득점값을 구한다(P: 득점, M: 측정치, Me: 중위수, Mu: 최대값, M_L : 최소값).

$$P = 50 + \frac{50 * (M - Me)}{(Mu - Me)} \qquad (식\ 1)$$

2) Positive 메트릭스의 하한값

Positive 메트릭스의 측정치 중 중위수보다 작은 측정치에 대해서는 (식 2)에 의해 득점값을 구한다.

$$P = \frac{50 * (M - M_L)}{(Me - M_L)} \qquad (식\ 2)$$

3) Negative 메트릭스의 상한값

Negative 메트릭스의 측정치 중 중위수보다 큰 측정치에 대해서는 (식 3)에 의해 득점값을 구한다.

$$P = 50 - \frac{50 * (M - Me)}{(Mu - Me)} \qquad (식\ 3)$$

4) Negative 메트릭스의 하한값

Negative 메트릭스의 측정치 중 중위수보다 작은 측정치에 대해서는 (식 4)에 의해 득점값을 구한다.

$$P = 100 - \frac{50 * (M - M_L)}{(Me - M_L)} \qquad (식\ 4)$$

5.3 계산값의 득점화

모델의 득점 리스트는 메트릭스에 대해 구해진 계산값 리스트를 (식 1)~(식 4)에 의하여 비율척도로 나타낸 값이다.

ID	1) ST	2) RA	3) SZ	4) LT	5) VM	6) LV	7) DS	8) MC
MODULE/FN1.bin	93.20	10.40	78.18	82.83	87.01	87.27	96.06	89.47
MODULE/FN2.bin	96.60	34.01	100.00	100.00	100.00	24.65	100.00	100.00
MODULE/FN3.bin	92.23	26.98	85.45	92.70	94.79	42.52	99.55	100.00
MODULE/FN4.bin	95.15	19.84	85.45	91.42	93.88	70.10	99.02	100.00
MODULE/CF4.bin	100.00	0.00	96.36	98.71	99.24	8.48	99.97	100.00
MODULE/CF5.bin	100.00	0.00	96.36	98.71	99.24	8.48	99.97	100.00
MODULE/CF6.bin	100.00	0.00	94.55	99.14	99.32	32.73	99.94	100.00
MODULE/CF7.bin	100.00	0.00	94.55	99.14	99.32	32.73	99.94	100.00

(그림 11) 모델의 득점 리스트
(Fig. 11) Score list of model

6. 소프트웨어 품질관리 도구의 구현

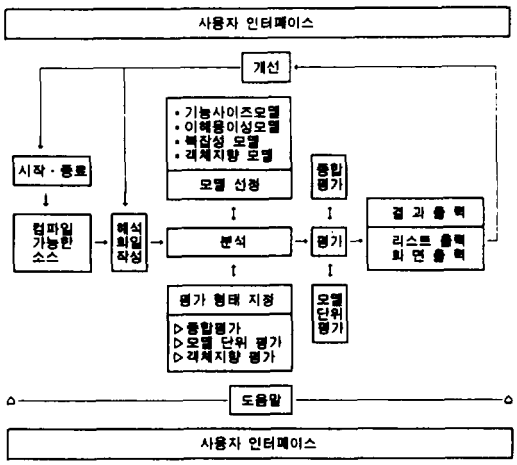
6.1 소프트웨어 품질관리 시스템

6.1.1 품질관리 시스템의 구조

구현한 소프트웨어 품질관리 도구는 소프트웨어 생명주기 중 프로그래밍, 테스팅, 유지보수 부분에 해당된다. 본 품질관리 도구의 목표는 소프트웨어의 품질을 테스트하고 보수성있는 모듈과 일양한 도구 작성을 위한 체계를 정립하고, 일관성있는 통합 환경을 제공함으로써 생명주기의 유지보수 비용과 노력을 감소하는 것이다.

구현한 소프트웨어 품질관리 도구는 원시 프로그램을 구문분석(lexical analysis)하여 해석화일을 작성하고, 이 해석화일을 토대로 하여 지정한 모델별 또는 전체 모델들에 대한 측정치를 도출한다. 이들 측정치를 이용하여 모델별로 구성된 메트릭스치를 추출하여 소프트웨어의 종합적인 품질 요소를 측정하게 된다.

원시 프로그램의 분석 결과와 득점 상황에 대해서는 품질평가의 효율적인 이해를 돕기 위해 가시적인 형태로 화면에 표시하고, 각각의 측정치에 대해서는 계산값 리스트, 득점 리스트, 종합지표 리스트, 모델 득점 리스트 등으로 출력하게 된다. (그림 12)는 제안한 도구의 구성도를 나타낸 것이다.



(그림 12) 소프트웨어 품질관리 도구의 구성도
(Fig. 12) Organization of software quality management tool

6.1.2 구현 환경

1) 적용 기종
- O·S: HP-UX
- X-Window 이용 가능한 환경

2) 작업 환경 및 소프트웨어
- HP-UX 9000/712
- X-Window X11R5
- X-Motif V1.2
- FLEX
- BYACC

3) 분석 프로그램 언어
- C 언어 및 C++로 작성한 소스 프로그램

6.2 품질평가 결과의 가시화

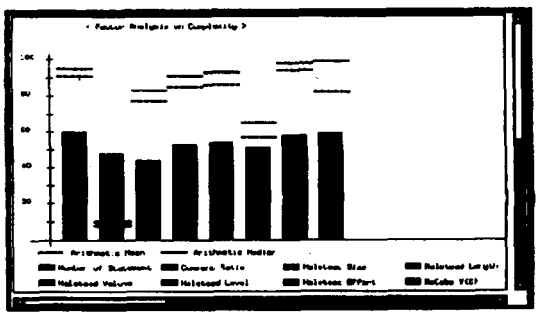
6.2.1 메트릭스 그래프

모델의 메트릭스 그래프는 모델의 메트릭스에 대한 득점값을 클래스의 메소드와 함수의 단위에서 측정해서 그래프 형태로 출력한다.

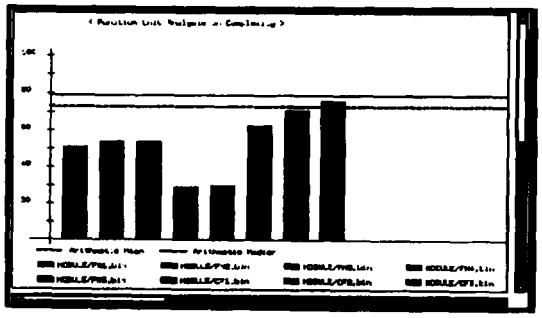
그래프에서 히스토그램 위의 점선과 실선은 각 메트릭스에 대한 평균값(mean)과 중앙값(median)으로서 해당 소프트웨어에 대한 득점 상황을 메트릭스의 평균과 중위수에 기준해서 파악할 수 있다.

6.2.2 함수 그래프

모델의 함수 그래프는 테스트되는 소프트웨어에 포함되어 있는 클래스 메소드(method)와 일반함수(free-subprogram)에 대한 함수별 득점값을 막대그래프로 표시한 것으로 하나의 소프트웨



(그림 13) 복잡성 모델의 메트릭스 그래프
(Fig. 13) Metrics graph of complexity model



(그림 14) 복잡성 모델의 함수 그래프
(Fig. 14) Function graph of complexity model

어에서 함수의 득점 상황과 다른 함수간의 득점을 비교해 볼 수 있다. 함수 그래프에 표시된 점선과 실선은 테스트된 함수의 평균값과 중앙값을 의미하는 것으로 함수별 득점이 평균값과 중앙값에 대해 어느 정도 양상을 띠고 있는가를 파악할 수 있다.

7. 결론 및 향후 연구 과제

소프트웨어는 요구 정의로부터 개발 종료에 이르기까지 제반 사항을 구현하는 것도 중요하지만, 개발이 완료된 이후의 유지보수도 매우 중요하다. 일반적으로 소프트웨어 생명주기 전 단계에서 소비되는 비용 중 유지보수에 관련된 비용이 80%에 이르고 있다. 따라서 유지보수성이 높은 고품질의 소프트웨어를 개발하기 위한 소프트웨어의 품질관리 도구를 개발하여 소프트웨어 품질에 대한 평가를 실시함으로써 품질을 향상시키고 소프트웨어 개발 비용을 절감할 수 있다.

본 논문은 소프트웨어의 품질을 관리할 수 있는 품질평가 체계와 자동화 도구의 개발에 관한 것으로서 유지보수 비용을 줄이고, 고 신뢰성과 고품질을 지닌 소프트웨어를 개발하는 방안으로 소프트웨어 개발 과정에서 원시 프로그램의 품질에 영향을 미치는 요인항목을 정량적으로 평가할 수 있는 소프트웨어 품질관리 체계와 자동화 도구의 개발을 목적으로 하였다.

소프트웨어 품질관리 도구의 개발은 다음과 같은 단계에 의해 구현하였다.

첫째, 품질관리 모델과 메트릭스를 이용하여 품질을 계측하고 테스트를 행하기 위해서 입력 데이타인 C 또는 C++ 언어로 이루어진 소스 프로그램을 분석하기 위한 파서(parser)를 구현하였다. 그리고, 파서를 통해 분석 화일 및 테스트 단위 화일을 함수 및 클래스 단위로 추출하였다.

둘째, 파서를 통해 추출된 결과와 테스트 단위 화일을 이용하여 메트릭스 값의 도출 및 분석에 이용하였다.

셋째, 평가 결과를 가시화하여 분석 결과를 직관적이고 명확하게 파악할 수 있도록 하는 사용자 인터페이스(user interface)를 구현하였다.

넷째, 품질관리 모델, 파서, 운용 루틴을 결합하여 소프트웨어 품질관리 도구의 내부 분석 시스템을 개발하였고 사용자 인터페이스를 연결하여 소프트웨어 품질관리 프로토타입(prototype) 도구를 구현하였다.

향후 과제로는 품질관리 도구에 실제 프로젝트 개발 방법을 통한 실용적이고 경험적인 분석(empirical analysis)을 이용한 가중치를 부여하여 평가치의 신뢰도를 높이는 연구와 출력 결과를 보다 직관적으로 출력할 수 있는 가시화에 대한 연구가 요구되며, 분석 중에 산출되는 화일 및 모듈을 체계적으로 관리할 수 있는 저장모델(repository)을 고려하여 품질관리와 아울러 프로그램의 이해도를 높이는데 연구가 진행되어야 할 것이다. 끝으로 안정된 소프트웨어 품질평가, 품질관리 도구의 구현을 위하여 지속적인 연구를 행하는 것이 요구된다.

참 고 문 헌

[1] Albreht, A. J. and Caffney, J. E., "Software Function, Source Line of Code, and Development Effort Prediction : A Software Science Validation", IEEE Trans. on SE, Vol. SE-9, No. 6, pp. 693-647, Nov. 1983.

[2] Booch, G., "Object-Oriented Analysis and Design with Application", The Benjamin/Cummings Publishing Company, Inc, 1994.

[3] Cheatham and Mellinger, "Testing Object Oriented Software Systems", Proc. of ACM SCS Con., 1990.

[4] Chidamber, S.R. and Kemerer, C.F., "To-

wards a Metrics suite for Object Oriented Design", OOPSLA '91, pp. 197-211, Oct. 1991.

[5] Conte, S.D., Dunsmore, H. E. and Shen, V. Y., "Software Engineering Metrics and Models", Benjamin/Cummings Publishing Company, Inc., 1985.

[6] Daskalantonakis, M.K., IEEE Member, "A Practical View of Software Measurement and Implementation Experiences Within Motorola", IEEE Trans. Software Eng., Vol. 18, No. 11, pp. 998-1010, Nov. 1992.

[7] DeMarco, T, "Controlling Software Projects: Management, Measurement and Estimation", Englewood Cliffs, Prentice Hall, 1982.

[8] Dumke, R., Neumann, K. and Stoeffler, K., "The Metric Based Compiler-A Concurrent Requirement", ACM SIGPLAN Notices, Vol. 27, No. 12, pp. 29-38, Dec. 1992.

[9] Fenton, N.E., "Software Metrics A Rigorous approach", CHAPMAN & HALL Publishing Company, Inc, 1991.

[10] Halstead, M.H, "Elements of Software Science", New York: Elsevier North Holland, 1977.

[11] Henderson-Sellers, B., "Some Metrics for Object Oriented Software Engineering", Technology of Object Oriented Languages and Systems Pacific, 1991.

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

[13] Jenson, R.L. and Bartley, J.W., "Parametric Estimation of Programming Effort: An Object-Oriented Model," The Journal of Systems and Software, Vol. 15, No. 2, pp. 107-114, May 1991.

[14] McCabe, T.J., "A Complexity Measure", IEEE Trans. on SE, Vol. SE-2, pp. 308-320, Dec. 1976.

[15] Moreau, D.R., and Dominick, W.D., "A Programming Environment Evaluation Methodology for Object-Oriented Systems: Part I -The Methodology", Journal of Object-Oriented Programming, pp. 23-32, Sep./Oct. 1990.

[16] Moreau, D.R., and Dominick, W. D. "A Programming Environment Evaluation Methodology for Application", Journal of Object-Oriented Programming, pp. 23-32, Sep./Oct. 1990.

[17] Pressman, R.S., "Software Engineering A Practitioner's Approach 3rd Ed.", McGraw-hill Publishing Company, Inc, 1992.

[18] 양해술 외, "소프트웨어 품질평가 체계와 자동화 도구의 개발", 한국통신연구개발원 장기 기초연구 보고서, 강원대학교, Sep. 1994.

[19] 한규정, "객체지향 프로그램의 품질평가 기준과 테스팅 속성에 관한 연구", 중앙대학교, 박사학위 논문, 1991.

[20] 조영식, 양해술 외, "소프트웨어 품질평가도구 (ESCORT)의 품질평가 체계", 정보과학회, '93 가을 학술발표 논문집, pp. 871-874, Oct. 1993.

[21] 권기현, 양해술 외, "소프트웨어 품질평가도구(ESCORT)의 내부 시스템 구현", 정보처리응용학회, '94 추계 학술발표 논문집, pp. 137-140, Oct. 1994.

[22] 이하용, 양해술 외, "소프트웨어 품질평가도구, ESCORT의 구현(II)", 정보과학회, '94 봄 학술발표 논문집, pp. 665-669, Apr. 1994.

[23] 김명옥, 양해술 외, "소프트웨어 품질평가도구 지원을위한 정량적인 측정모델", 정보과학회, '93 가을 학술발표 논문집, pp. 867-870, Oct. 1993.

[24] 양해술의 외, "소프트웨어 품질보증과 품질평가 자동화도구의 개발" 한국통신장기기초연구 1차년도 최종보고서, 강원대학교, Sep. 1993.

양 해 술

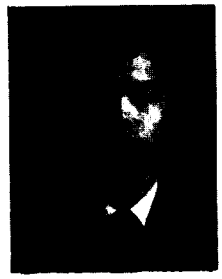
1975년 홍익대학교 공과 대학 전기공학과 졸업(학사)
1978년 성균관대학교 정보처리 학과 정보처리 전공(석사)
1991년 日本 오사카대학 기초 공학부 정보공학과 소프트웨어공학 전공(공학박사)
1975년~79년 육군중앙경리단 전자계산실 시스템분석장교 근무
1984년~92년 성균관대학교 경영대학원 강사
1986년~87년 日本 오사카대학 객원연구원
1993년~94년 한국정보과학회 학회지 편집부위원장
1994년~현재 한국산업표준원(IIS) 이사
1994년~현재 한국정보처리학회 논문지편집위원장
1980년~현재 강원대학교 전자계산학과 교수
관심분야 : 소프트웨어 공학(특히, S/W 품질보증과 평가, SA/SD, OOA/OOD/OOP, CASE, SI), 소프트웨어 프로젝트관리.

박 정 호

1980년 성균관대학교 사범대학 졸업(문학사)
1980년~82년 성균관대학교 경영대학원 정보처리 학과(경영학석사)
1985년~87년 日本 오사까대학교 대학원 정보공학전공(공학석사)
1987년~90년 日本 오사까대학교 대학원 정보공학전공(공학박사)
1994년~현재 한국정보처리학회 학회지 편집위원장
1991년~현재 선문대학교 전자계산학과 조교수
관심분야 : 분산 알고리즘, 소프트웨어공학.

권 기 현

1993년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)
1995년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)
1995년~현재 강원대학교 대학원 전산학과 박사과정
관심분야 : 소프트웨어공학(특히, S/W 품질보증과 품질평가, 객체지향 프로그래밍, 객체지향 분석과 설계, CASE).



이 하 용

1993년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)
1995년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)
1995년~현재 강원대학교 대학원 전산학과 박사과정
관심분야 : 소프트웨어공학(특히, S/W 품질보증과 품질평가, 객체지향 프로그래밍, 객체지향 분석과 설계, CASE).

조 영 식

1992년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)
1994년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)
1995년~현재 강원대학교 대학원 전산학과 박사과정
1995년~현재 상지전문대학 전자계산과 강사
관심분야 : 소프트웨어공학(특히, S/W 품질보증과 품질평가, S/W 신뢰도 및 비용산정, 객체지향 분석과 설계).

이 용 근

1988년 강원대학교 자연과학대학 전자계산학과 졸업(이학사)
1994년 강원대학교 전자계산학과 소프트웨어공학 전공(이학석사)
1989년~92년 강원대학교 전자계산학과 조교
1995년~현재 강원대학교 대학원 전산학과 박사과정
1994년~현재 한림전문대학 강사
관심분야 : 소프트웨어공학(특히, S/W 품질보증과 품질평가, 시스템통합 및 분산개발환경, 객체지향 분석과 설계).

허 태 경

1983년 경북대학교 전자공학과 졸업(학사)
1985년 경북대학교 대학원 전자공학과 졸업(공학석사)
1985년 경북대학교 공과대학 조교
1986년~94년 한국통신 품질보증단 S/W 품질연구부장
1994년~현재 한국통신 품질보증단 개발품질 부장
관심분야 : 소프트웨어공학(특히, 소프트웨어 품질보증과 품질평가의 체계, 인증, 표준화).