재사용가능한 클래스 후보자들의 품질 메트릭들에 관한 연구

김 재 생† · 송 영 재††

요 약

소프트웨어 품질 평가는 개발 중인 시스템이나 완성된 시스템의 품질을 평가하여 문제점을 수정하고, 기존의 원시코드에서 재사용가능한 부품을 추출하는 등 여러 연구 분야에 이용되어왔다. 본 연구에서는 구현한 재사용 검색 시스템(KHR System)[11]을 통해 나온 클래스 후보자들에게 재사용성 여부를 측정할 수 있는 품질 평가에 대한 객관적인 추정 함수들을 제안, 적용하여 가장 적절한 후보자를 선택하도록 하였다. 이 논문에서 제안한 클래스 후보자들에 대한 정량적인 품질 측정은 재사용하고자 하는 후보자 클래스들을 비교 평가할 수 있다는 장점을 가지고 있다.

A Study on Quality Metrics of Reusable Classes Candidates

Jae Saeng Kim† · Young Jae Song††

ABSTRACT

It is used in many researches that the s/w quality evaluation evaluates the developing system or the developed system, updates the problems and selects the reusable components from source code. In this paper, we propose the objective metric functions which can evaluate the reusability of candidates classes with the KHR system[11] and select a proper candidate. The quantitative quality metrics we proposed have merits to compare and to evaluate the reusable candidates classes.

1. 서 론

소프트웨어의 품질 평가는 여러가지 대안 소프트웨어들 중에서 가장 좋은 시스템을 선택하기 위해서, 사용자의 요구사항과 목표 시스템을 비교하기 위해서, 후보자가 과연 만족스러운 시스템인지 판단하기 위해서 행하여졌다. 평가 과정[1]은 소프트웨어 품질의 측정, 등급화(rating), 평가(assesment)의 3단계로 나눌 수 있다. 소프트웨어 품질 측정은 부품들의 품질을 계산하여 나온 값에 의해 등급화되어 평가된다.

재사용가능한 부품들의 원시코드들은 대부분 절차언어와 객체지향언어로 구성되어 있다. 절차언어의 품질 메트릭의 종류들은 프로그램 크기에 의한 Halstead의 노력도 산출[2], McCabe의 순환수 산출[2], 기능점 분석, 데이터 양 메트릭스와 정보흐름 메트릭스 등[15]이 있으나 이들 품질 메트릭들이 operand와 operator에만 치중하여 객체지향언어의 재사용가능한 후보자인 클래스들에게는 유용하지 못하다는 단점을 갖고 있다.

객체지향언어에서는 Chidamber과 Kemerer[3]가 메

† 정 회 원:경희대학교 전자계산공학과
†† 종신회원:경희대학교 공과대학장
논문접수:1996년 5월 2일, 심사완료:1997년 1월 21일

소드들의 수, 클래스의 깊이, 칠드런의 수 등을 가지고 클래스 품질을 측정하였다. 또한, Binder[4]는 클래스의 성질을 캡슐화, 상속, 계층성, 복잡도를 사용해서 테스트할 수 있음을 이론적으로 설명했지만, 정량적인 평가에 관한 방법은 실제로 실행하지 못했다.

이와 같이 소프트웨어 시스템의 품질 평가는 현재 많이 연구되고 있지만 재사용가능한 부품인 클래스들의 정량적인 품질 평가에 관한 연구는 아직도 미흡한 단계이다.

본 논문에서는 객체지향언어인 C++ 언어의 재사용가능한 클래스들의 품질들을 정량적으로 계산할 수 있는 메트릭 함수들을 정의하고, 정의식에 의해 나온 값들에 의해 재순서화된 리스트 중에서 가장 적절한 재사용가능한 후보자를 선택할 수 있는 평가 지원 시스템을 KHR 재사용 시스템[11]에 부가하였다.

2. 관련개념

2.1 기존의 재사용 시스템

이전에는 사용자가 검색된 후보자 부품들을 수동작으로 평가했으나 소프트웨어 라이브러리가 커짐에 따라 재사용가능 부품들을 자동으로 평가하는 툴을 원하게 되었다.

RSL 시스템[5]은 주로 ADA언어를 지원하는 시스템으로서, 라인 카운트, 복잡도, 가독성(readability), 프로그램 구조와 스타일, 문서화 정도 등을 사용하여 후보자들을 평가했다. 이 시스템의 특징은 "barometer"를 사용하여 메트릭의 중요성 정도를 사용자가 직접 입력하도록 하였다. 그러나 사용자가 그 메트릭의 중요성 정도를 입력하여야 하므로 소프트웨어 공학 개념을 이해하는 사용자들만이 사용할 수 있다는 단점이 있다.

Prieto Diaz와 Freeman의 시스템[8]은 후보자들을 평가하는 척도로서 프로그램 크기, 프로그램 구조, 프로그래밍 언어, 프로그래머의 숙련도, 문서화 정도 등의 5가지 품질 속성을 사용했다. 이 품질들의 값들은 퍼지이론을 사용해서 평가했으며, 정량적으로 측정하지 못하였고 숙련가가 이 품질의 값들을 임의로 부여했다는 단점을 갖고 있다.

CARE 시스템[14]은 C 언어를 지원하는 시스템으로 모듈의 크기, McCabe의 복잡도, 정규성, 재사용 빈도수 등을 사용하여 후보자들을 평가했다. 모듈의 크기는 Halstead의 소프트웨어 과학을 사용했으며, 정규성은 실제 크기와 추정치의 차이정도를 구하였다. 재사용 빈도수는 사용자가 정의한 모듈의 호출 정도와 표준함수의 평균 호출 정도를 비교하였다. 기존의 시스템보다는 정량적으로 품질의 값을 추출할 수 있지만, 절차언어의 속성에만 치중하여 객체지향언어의 속성을 반영하지 못하였다.

CARS 시스템[9]은 후보자의 크기, 코드의 표준화, 이식성, 응집도, 결합도, 복잡도 등을 사용하여 후보자들을 평가했다. 시스템의 환경면에서는 절차언어와 객체지향언어를 지원하지만, 후보자 평가면에서는 절차언어의 속성에만 치중되어 객체지향언어의 속성들의 값을 비교할 수가 없었다.

2.2 기존의 품질 메트릭

절차언어를 사용하여 구성된 품질 메트릭들은 다음과 같은 여러가지 방법들이 있다. 코드 라인수는 블랭크 라인, 코멘트, 선언문, 널(NULL) 문장을 카운트하지 않았다. Halstead의 노력도[15]는 알고리즘을 구현하는 데 요구되는 프로그램의 난이도 정도를 말한다. McCabe의 순환수[15]는 모듈의 제어흐름의 복잡도를 뜻한다. 순환수는 $v(g)=e-n+2$, e=edges의 수, n=nodes의 수로 구할 수 있으며 V(G)의 값이 10보다 크면, 모듈을 분해하는 것이 좋다. Caldiera와 V. R. Basili[14]는 McCabe의 순환수, 정규성, 재사용 빈도수, Halstead의 볼륨(volume)을 사용했다. 문서화 정도[7]는 모듈의 정확한 기능, 용법과 인터페이스를 정확히 사용자에게 알려준다. 원시코드 대 코멘트의 비율은 50:50이 좋다.

객체지향언어에서, McCabe의 복잡도는 별로 의미가 없다. R.V.Binder[4]는 캡슐화, 상속도, 복잡도를 품질로서 제안했다. 캡슐화는 메소드의 비응집도, public 부와 protected 부의 데이터 멤버들의 퍼센트와 액세스의 수를 구하여 클래스들간의 부작용을 테스트했다. 상속성은 클래스 계층의 수, 파생 클래스의 수, 상속 트리의 깊이, 폴리모피즘 등을 사용해서 측정하였다. 복잡도는 McCabe의 순환수, 객체들간의 결합도 등을 구하여 측정했다. I.Jackson[2]은 process 메트릭으로 전체 개발시간, 수정 소비시간, faults의 수, 품질 보증 비용, 개발과정과 툴을 소개하는 비용 등을

사용하고, product related 메트릭으로 클래스들의 수, 재사용된 클래스들의 수, 상속성의 높이, 깊이를 사용하고, 통계 메트릭으로는 한 클래스의 operations의 수, operation의 길이, 상속된 operation의 평균수 등을 사용해야 한다고 주장했다. Chidamber와 Kemerer[3]는 평가 척도로서 메소드들의 수, 상속 트리의 깊이, 칠드런의 수, 결합도, 클래스에 대한 응답, 응집도 등을 제안했다.

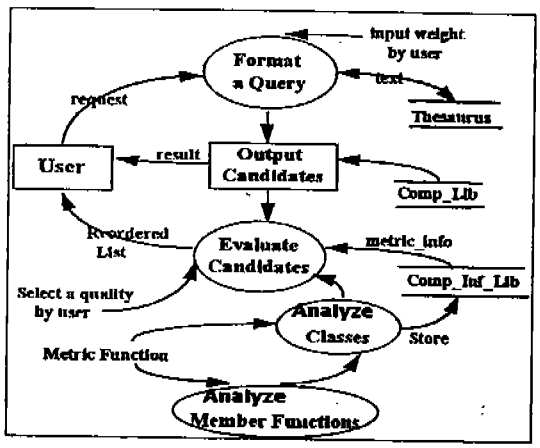
이상의 연구 논문들에서, 절차언어의 품질들은 객체지향언어의 품질 속성을 반영하지 못하였고, 객체지향언어의 품질들은 이론에만 치중하여 값들을 정량적으로 추론하지 못하였다.

본 논문에서는 절차언어와 객체지향언어의 품질 속성들 중 재사용성과 관련된 품질들인 클래스의 크기, 복잡도, 정보은닉도, 상속도, 확장성, 문서화 정도 등의 6개 품질 메트릭들을 정량화된 식으로 정의하고, 제안한 정의식들을 2개 화일에 적용하여 테스트했다.

3. 후보자 평가를 위한 지원환경

3.1 시스템의 구성도

KHR시스템[11]은 질의 과정, 후보자 출력, 후보자 평가 과정을 거쳐 가장 적절한 후보자가 나오는 사용자 인터페이스 부분을 설계, 구현하였다. 그러나 후보자 평가를 위해 사용된 속성값들은 수동작으로 계산된 값들을 이용하였고, 자동화과정은 현재 구현중이다.



(그림 1) 후보자 평가서브시스템의 DFD
(Fig. 1) DFD of candidates evaluation subsystem

(그림 1)은 본 논문에서 구현한 후보자 평가 서브시스템의 자료흐름도이다.

사용자가 그래픽 환경에서 브라우징 기법을 사용하여 텍스트와 weight를 입력하면 질의어가 구성된다. 검색 시스템은 텍스트와 weight로 구성된 질의어를 번역하고, 부품 정보 라이브러리를 검색해서 질의어와 유사한 후보자들을 발생한다. 재사용자는 비교하고자 하는 품질 메트릭을 마우스를 사용하여 한 개 선택한다. 그 결과 선택된 품질 메트릭에 따라 재순서화된 후보자 리스트가 사용자에게 디스플레이된다.

후보자 평가 서브 시스템은 UNIX상에서 Motif 환경과 C 언어를 사용하여 구현하였고, KHR 재사용 시스템에 부가하였다[11].

3.2 클래스의 품질 평가 메트릭들과 지원환경

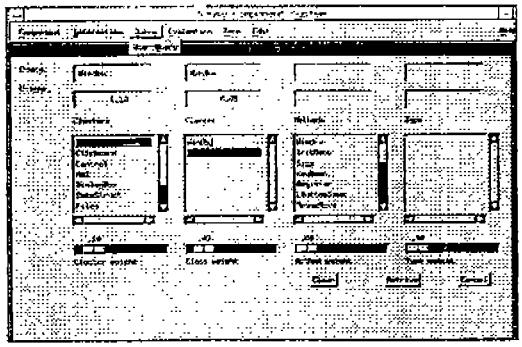
메트릭들은 보통 개발 중인 시스템이나 완성된 시스템의 품질을 평가하는 데 사용되었고[1][6], 또 일부 분야에서는 기존의 코드에서 재사용가능한 부품을 추출할 수 있는 품질 측정에 사용되어 왔다[14]. 그러나 본 논문에서는 질의결과 나온 여러개의 유사한 후보자들에게 우리가 제안한 품질 메트릭들을 적용하고, 메트릭의 종류에 따라 후보자들의 값을 비교하였다.

한 개의 클래스는 유사한 객체들의 속성과 메소드들을 정의한 것이므로, 절차언어면과 객체지향언어 속성면을 모두 고려하여야 한다. 한 클래스내에 메소드들은 절차언어의 속성을 가지므로 Size, Effort, McCabe의 순환수, 관계도, 문서화정도 등을 속성으로 들 수 있으며, 객체지향면의 클래스의 속성들은 데이타 추상화 기능인 캡슐화, 자료은닉, 상속성, 다형성과 오버로딩 등의 특징들을 갖고 있다. 이 속성들은 객체지향 정도와 재사용 정도를 측정하는 데 도움을 준다[8][16]. 그러므로 정량적인 클래스들의 속성값들에 따라 후보자들을 평가할 수 있다.

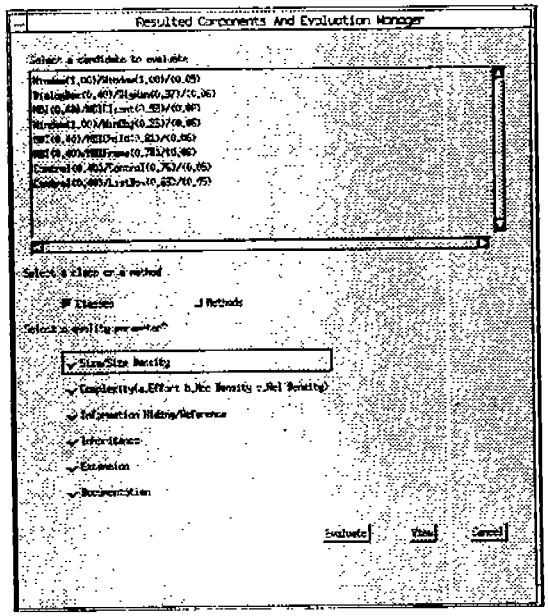
(그림 2)는 4개의 브라우저에서 텍스트를 선택하여 질의어를 형성하는 과정이다. Cluster 명은 응용분야를, Class 명은 찾고자하는 클래스명을 해당 브라우저에서 마우스를 사용하여 선택한다. Weight는 찾고자 하는 부품의 유사한 정도를 나타낸다. 따라서, weight를 안주면 단 한 개의 부품을 검색하지만, weight를 주면 유사한 정도의 부품들을 weight 범위안에서 모

주 찾아주는 장점이 있다.

(그림 3)은 (그림 2)의 질의 결과 나온 여러 후보자들과 각 후보자들의 품질에 따라 후보자들을 비교할 수 있는 메트릭의 종류들을 보여주고 있다. 이 품질 메트릭들의 자세한 설명은 다음절에서 설명하였다.



(그림 2) 클래스 질의 화면
(Fig. 2) Query screen for classes



(그림 3) 재사용가능한 후보자들과 품질 선택 과정
(Fig. 3) Rusable candidates and the process of quality selection

3.2.1 클래스 크기

클래스 크기는 재사용 비용과 메트릭에 영향을 준다. 객체지향언어의 멤버함수 크기[17]는 보통 사람이 한눈에 볼 수 있는 한 페이지 정도가 좋으며, 실제 크기는 30라인 정도이다. [정의 1]식에서 한 클래스의 크기(SC)는 클래스에서 멤버 데이터와 멤버함수들의 선언 문장수와 멤버함수들의 실제 문장 라인수를 더했는데, 그 이유는 클래스의 실제 크기를 계산하기 위해서이다. 클래스의 크기위반정도(SDC)는 한 클래스에서 총 멤버함수들의 수에 대하여 30라인이 넘는 멤버함수들의 수를 구하였다.

[정의 1] 한 클래스의 크기 및 크기위반정도

$SC(Size\ of\ Class)$ = 멤버들과 멤버함수들의 선언문장수

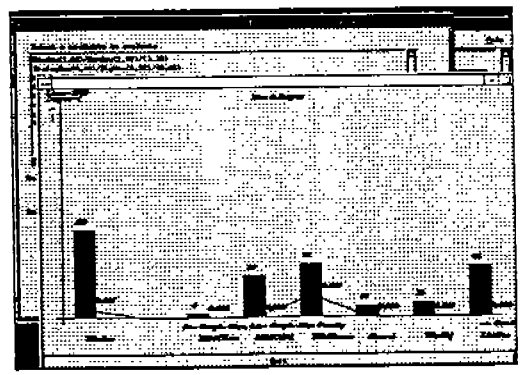
$$+\sum_{i=1}^{n} SM_i$$

$SM_i(Size\ of\ Function)$ = i번째 멤버함수의 크기

$SDC(Size\ Density\ of\ Class)$ =

$$\frac{\text{30 라인이 넘는 멤버함수들의 수}}{\text{총 멤버함수들의 수}}$$

(그림 4)는 사용자가 후보자들을 비교하기 위해 (그림 3)에서 Size 버튼을 선택했을 때, 클래스들의 크기와 크기위반정도에 따른 결과를 나타낸다. 크기위반정도가 1이면, 가장 나쁜 클래스후보자이므로 대부분의 후보자 클래스들이 크기 위반정도가 낮다는 것을 알 수 있다.



(그림 4) 후보자들의 크기와 크기위반정도
(Fig. 4) Size vs. size density of candidates

3.2.2 복잡도

복잡도는 Halstead의 노력 방정식과 McCabe의 순환수[15]를 이용하여 한 클래스의 복잡도 정도 구할 수 있다. McCabe의 순환수(v(g))의 값은 10보다 크면, 모듈은 분해하는 것이 좋으므로, 한 클래스에서의 순환수 위반정도를 구할 수 있다. 노력도와 McCabe의 순환수는 너무 크면 모듈이 복잡하고 이해하기 어려우므로 재사용이 곤란하다. 본 논문에서는 한 클래스의 노력도(EC)와 순환수(CNC)는 [정의 2식]과 같이 한 클래스에서 정의된 멤버함수들의 노력도와 순환수를 더한 값을 구하였다. 또한, 한 클래스에서 정의된 멤버함수들 중에서 McCabe의 순환수(v(g))가 10이 넘는 멤버함수들의 수를 구하여 순환수의 위반정도(DCN)를 구한다.

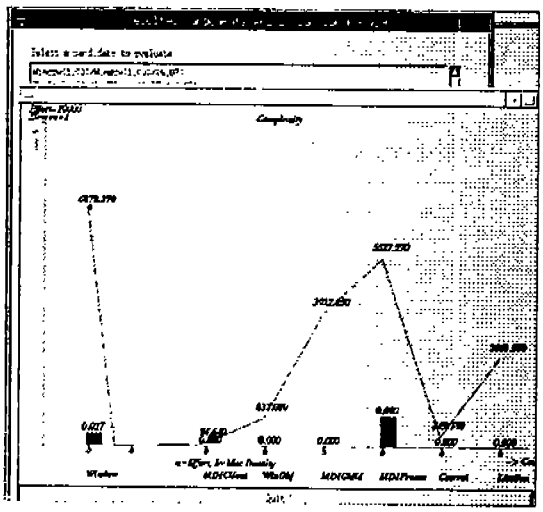
[정의 2] 한 클래스의 노력도와 순환수

$$EC(Effort\ of\ Class)=\sum_{i=1}^{n} EM_i,$$

$EM_i = i$번째 멤버함수의 노력도

$$CNC(Cyclomatic\ Nmber\ of\ Class)=\sum_{i=1}^{n} CM_i,$$

$CM_i = i$번째 멤버함수의 순환수



(그림 5) 후보자들의 노력도와 순환수 위반정도
(Fig. 5) Effort and cyclomatic density of candidates

$$DCN(Density\ of\ Cyclomatic\ Nmber)=\frac{V(G)\text{의 값이 10이 넘는 멤버함수들의 수}}{\text{멤버함수들의 총 수}}$$

(그림 5)는 후보자들을 비교하기 위하여 (그림 3)에서 Complexity 버튼을 선택했을 때의 결과이다. 실선 그래프는 노력도 값이며, 막대 그래프는 순환수 위반정도를 나타내고 있다. 노력도면에서는 Window 클래스가 가장 높았고, McCabe의 순환수 위반정도가 높은 것은 MDIFrame 클래스였다.

3.2.3 정보 은닉도

정보은닉(information hiding)이란 캡슐화 속에 있는 항목에 대한 정보를 외부에 감추는 것이다. 예를 들면, private 부에 선언된 데이터는 클래스 외부의 함수가 접근할 수 없고, 그 클래스 내부에서만 접근이 가능하다. 또한, 파생 클래스의 객체들은 베이스 클래스의 private 부나 protected 부의 멤버들을 접근할 수 없다. 액세스된 수가 많으면 많을수록 그 멤버는 정보은닉이 잘된 것이다. 한 클래스의 캡슐화된 정도(EDC)는 [정의 3식]과 같이 정보은닉된 멤버들과 멤버함수들의 선언된 수에 비하여 정보은닉된 멤버들과 멤버함수들의 액세스된 총수를 구하였다.

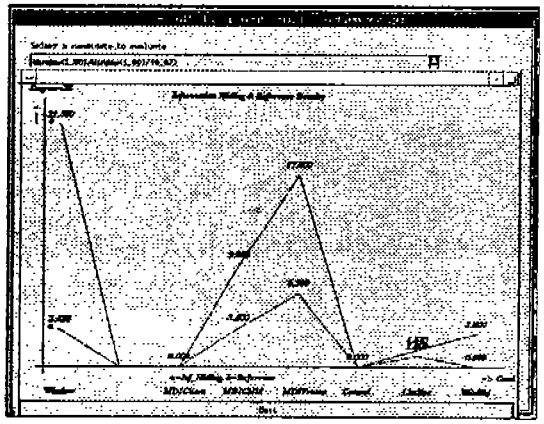
참조자(&)는 주소 연산자와 같지만, 같은 연산자는 아니며 객체에 대한 다른 이름으로 행세하므로, 많이 사용하면 할수록, 정보의 은닉이 좋다. 그러므로 한 클래스에서 참조자가 선언된 수에 비해 참조자가 사용된 총수를 구하여 한 클래스에서 참조자가 사용된 정도(RDC)를 구하였다.

본 절에서는 한 클래스의 캡슐화와 참조자 사용정도를 계산하여 정보은닉정도를 계산하였다.

[정의 3] 한 클래스의 정보 은닉 정도

$$EDC(Encapsulation\ Density\ of\ Class)=\frac{\text{액세스된 총수}}{\text{Private와 Protected내의 멤버 및 멤버함수들의 수}}$$

$$RDC(Reference\ Density\ of\ Class)=\frac{\text{참조자가 사용된 총 수}}{\text{한클래스에서 참조자가 선언된 수}}$$



(그림 6) 후보자들의 정보 은닉도
(Fig. 6) Information hiding density of candidates

(그림 6)은 (그림 3)의 Information Hiding/Reference 버튼을 선택했을 때의 결과이며, 정보 은닉면에서는 MDIFrame 클래스가, 참조자 사용면에서는 Window 클래스가 사용정도가 가장 높음을 알 수 있다.

3.2.4 상속도

파생 클래스는 베이스 클래스의 속성(data type)이나 멤버함수들을 물려받고 새로운 속성과 새기능들이 추가되기 때문에 상속도(inheritance)는 확장성과 재사용성을 지원한다.

파생의 형이 public인 경우에 파생 클래스는 베이스 클래스의 protected 부와 public 부의 멤버들을 액세스할 수 있고, 파생 클래스의 객체들은 베이스 클래스의 public 부의 멤버들만 접근 가능하므로 상속도는 IPC_1으로 정의했다.

파생의 형이 private인 경우에 파생 클래스는 베이스 클래스의 protected 부와 public 부의 멤버들을 접근할 수 있고, 파생 클래스의 객체들은 베이스 클래스의 모든 멤버들을 접근할 수 없으므로 상속도는 IPC_2로 정의하고 베이스 클래스의 상속도를 0 값으로 계산한다.

[정의 4] 한 클래스의 상속도

IBC(Inheritance density of Base Class) =

$$\frac{\textit{Protected}\text{와 }\textit{Public}\text{의 멤버 및 멤버함수들의 수}}{\text{베이스클래스의 멤버 및 멤버함수들의 총 수}}$$

IPC_1(Inheritance density of Public Class) =

$$\frac{\textit{Protected}\text{와 }\textit{Public}\text{의 멤버 및 멤버함수들의 수}}{\text{파생클래스의 멤버 및 멤버함수들의 총 수}} + \frac{\textit{Public}\text{의 멤버 및 멤버함수들의 수}}{\textit{BASE}\text{클래스의 멤버 및 멤버함수들의 총 수}}$$

IPC_2(Inheritance density of Private Class) =

$$\frac{\textit{Protected}\text{와 }\textit{Public}\text{의 멤버 및 멤버함수들의 수}}{\text{파생클래스의 멤버 및 멤버함수들의 총 수}}$$

DMI(Density of Multiple Inheritance) =

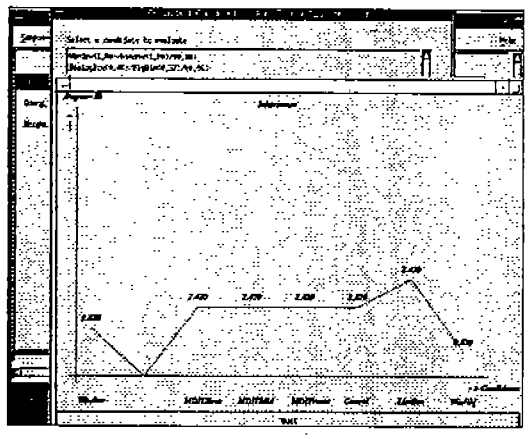
$$\frac{\textit{Protected}\text{와 }\textit{Public}\text{의 멤버 및 멤버함수들의 수}}{\text{파생클래스의 멤버 및 멤버함수들의 총 수}} + \sum_{i=1}^{n} IBC_i, \qquad i = \text{베이스 클래스의 수}$$

$$IDD(\textit{Inheritance Density by Depth}) = \sum_{i=1}^{n} IC_i,$$

i = 깊이의 수, ICi = i번째 깊이 클래스의 상속도

파생의 형이 protected일 경우에는 파생의 형이 private인 경우의 성질과 같으므로 IPC_2와 같이 계산한다. 다중상속일때는 부모 클래스가 2개 이상이므로 상속도는 DMI식과 같이 정의하였다.

Booch[20]는 상속계층성에서 깊이와 폭을 7±2 클래스들로 제한하였다. Chidamber[3]는 클래스의 계층



(그림 7) 후보자들의 상속도
(Fig. 7) Inheritance density of candidates

구조에서 폭보다는 깊이(depth)를 갖는 것이 상속을 통하여 재사용을 증진시킨다고 하였다. 이 논문에서는 부모의 수(DMI식)와 깊이에 의한 상속(IDD식)만을 고려하였다. 그러나 깊이가 5~6을 초과하면 멤버함수의 실행경로가 길어져 이해하는 데 어려움이 있으므로 재사용하기가 어렵다. 깊이가 2 이상일 경우의 상속도는 상위 레벨의 상속도를 그대로 이어받아 [정의 4식]의 IDD와 같이 계산하였다.

(그림 7)은 (그림 3)의 Inheritance 버튼을 선택했을 때의 결과이며, WinObj 클래스이외의 후보자들은 대체로 상속도가 2배 이상 높았다.

3.2.5 확장성

확장성(extension)[5][17]은 기존의 부품을 새로운 요구에 대응하기 위하여 쉽게 확장할 수 있는 성질이므로 재사용성과 연관이 있다.

다형성(polymophism)인 함수 오버로딩와 연산자 오버로딩은 한 함수명이 여러가지 다른 버전을 가지며, 가상함수는 베이스 클래스의 멤버함수를 파생 클래스에서 재정의할 수 있고, 프렌드 함수와 프렌드 클래스는 다른 클래스의 멤버들을 액세스하기 위해서 사용된다.

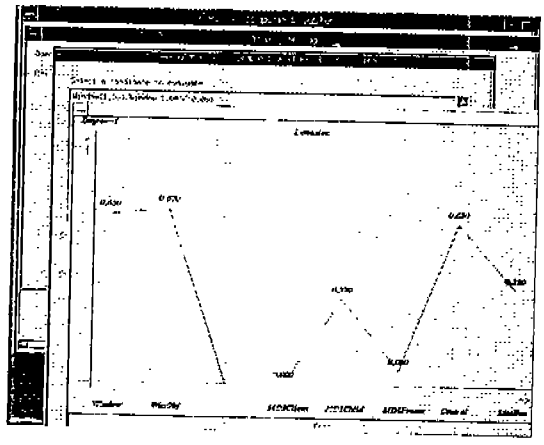
하나의 멤버함수를 이용할 때마다 함수호출이 발생하면 매우 비효율적이기 때문에, inline 함수와 macro 함수들은 함수호출 대신에 원시코드를 호출위치에 놓아 확장되어 처리되었다. 마크로 함수는 형을 보장하지 못하나 인라인 함수는 형을 보장한다. Inline 함수는 클래스 계층구조에 따라 새로이 요구되는 데이터형이나 기능을 쉽게 첨가할 수 있다.

본 논문에서는 [정의 5식]과 같이 한 클래스의 멤버함수의 총수에 비해 확장가능한 성질을 가진 함수들의 수를 구하여 확장성 정도를 구하였다.

[정의 5] 클래스의 확장성

EXT(EXTension) = (함수와 연산자 오버로딩 +가상함수 +프렌드함수 +*Inline*함수 +마크로함수들의 수) /(한 클래스의 멤버 및 멤버함수들의 총 수)

(그림 8)은 (그림 3)의 Extension 버튼을 선택했을 때의 결과이다. 확장성면에서 후보자들을 비교해볼 때 Window 클래스, Control 클래스, WinObj 클래스들이 좋은 후보자들이다는 것을 알 수 있다.



(그림 8) 후보자들의 확장성
(Fig. 8) Extension density of candidates

3.2.6 문서화

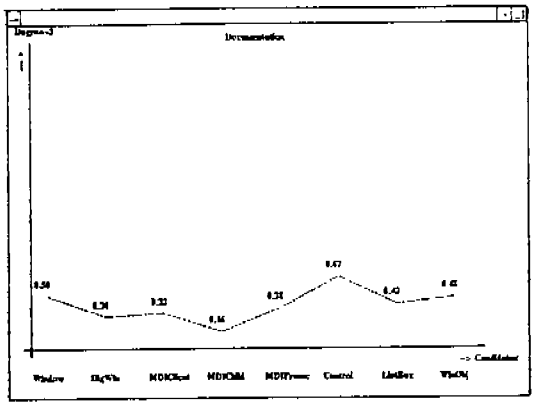
재사용에 있어서 모듈의 정확한 기능, 용법과 인터페이스를 정확하게 사용자에게 알려주는 것이 필요하다. 모듈내에 정확한 주석을 달고 암호적인 데이터 이름과 비구조적인 흐름을 사용하지 않는 것이 모듈의 재사용 가능성을 높여준다.

문서화(documentation)는 클래스내에서 코멘트 정도와 멤버 함수들의 구조적 단순성을 측정하여 정도를 구할 수 있다. 코멘트 비율[7]은 50:50이 좋으며, 구조적 단순성은 순차, 선택, 반복구조, 스타일의 일관성({, })문의 사용, 의미있는 데이터명이나 함수명의 사용들을 고려하였다. 모호한 코드나 goto 문장은 문장의 흐름을 읽기 어렵고 이해하기 어렵게 하므로 구조적 단순성에서 제외한다. 문서화 정도(DOC)는 [정의 6]식에서 한 클래스의 총라인수 중에서 코멘트 라인수와 구조적인 문장수가 차지하는 비율을 구하였다.

[정의 6] 클래스의 문서화 정도

$$DOC(DOCumentation) = \frac{\text{코멘트 라인수} + \text{구조적 단순성과 관련있는 라인수}}{\text{한 클래스의 총 라인수}}$$

(그림 9는) (그림 3)의 Documentation 버튼을 선택했을 때의 결과이다. 문서화 정도는 0.5 이상인 클래스들이 알맞은 후보자들이므로 Window 클래스와 Control 클래스가 문서화면에서는 가장 적절하다.



(그림 9) 후보자들의 문서화 정도
(Fig. 9) Documentation density of candidates

4. 실험 및 평가

우리들은 2 개의 화일[12][13]에서 클래스들을 추출한 다음, 3장에서 제안한 정의식에 의해 후보자 클래스들의 재사용가능한 품질을 수동작으로 분석하였다 〈표 1〉.

분석된 클래스들의 수는 93개, 멤버함수들의 수는 847개, 총 라인수는 6171라인, 한 클래스당 평균 함수의 수는 7-12개 이었다.

〈표 1〉 분석된 클래스들의 특성
〈Table 1〉 Characteristics of analyzed classes

화일명	클래스의 수	함수의 수	총라인수	한클래스당 평균함수의 수
Windows	58	422	2,786	7.285
CASE	35	425	3,385	12.142

한 클래스는 7±2개의 멤버함수들을 다루는 것이 좋고[18], 한 멤버함수의 크기는 한 페이지 크기인 30라인 정도가 좋기 때문에, 한 클래스의 실제크기는 210라인 이하인 것이 좋다. 클래스의 크기위반정도는 최대 1과 최소 0값을 기준으로 하였다.(3.2.1절 참조) 〈표 2〉에서 클래스들의 평균크기가 210라인보다 낮고, 크기위반정도의 값이 0.1이하이므로, 크기면에서는 좋은 클래스들이라는 것을 알 수 있다.

노력도는 알고리즘을 구현하는 데 드는 정신적인 비용을 말한다. 노력도는 크면 클수록 모듈이 복잡하고, 재사용 비용이 많이 든다는 것을 의미한다. 〈표 2〉에서 두 화일의 노력도가 높다는 것을 알 수 있다.

McCabe의 순환수는 모듈의 제어흐름 복잡도를 말하며, 그 값이 10이상이면 모듈을 분할하는 것이 좋다. 클래스의 순환수는 멤버함수들의 수에 영향을 받는다. 한 클래스는 7±2개의 멤버함수들을 갖는 것이 좋으므로, 한 클래스의 순환수 값은 90이하가 좋다. 순환수 위반정도는 최대 1값과 최소 0값을 기준으로 하였다. 〈표 2〉에서 두 화일의 순환수는 안정성이 있고, 순환수 위반 정도도 낮았다.

문서화정도는 0.5이상이 좋으므로(3.2.6절 참조), Window 화일보다 CASE 화일의 문서화 정도가 좋았다.

〈표 2〉 절차언어면의 품질 메트릭 비교
〈Table 2〉 Comparision of quality metrics in procedural language

	메트릭의 종류	화 일 명	
		Windows	CASE
Size	클래스 평균 크기(SC)	47.172	96.714
	크기위반정도(SDC)	0.016	0.031
노력도	클래스 평균 노력도(EC)	2666.436	10259.473
(Effort)	멤버함수 평균 노력도(EM)	313.882	966.042
McCabe 순환수	클래스 평균순환수(CNC)	15.224	31.2
	클래스 순환수 위반정도(DCN)	0.013	0.035
	멤버함수 평균 순환수	1.996	2.805
문서화	평균 문서화정도(DOC)	0.328	1

〈표 3〉은 객체지향언어면의 품질 메트릭에 관한 정량적인 측정 값이다.

정보은닉성은 멤버들과 멤버함수들과 참조자가 선언된 수에 비하여 액세스된 수가 많으면 많을 수록, 그 멤버는 정보은닉이 잘된 것이다(3.2.3절 참조). 정보은닉성과 참조자 사용정도가 높으면 높을수록, 정보은닉이 잘된 클래스이므로, 두 화일의 정보은닉성

이 좋다는 것을 알 수 있었다.

상속도는 깊이에 따라 재사용을 증진시키는 데, 깊이가 2~3인 것이 재사용하기가 쉽고 깊이가 5이상이면 이해하기가 어려워서 재사용하기가 어렵다. 그러므로 깊이가 깊고 상속도가 좋다고해서 재사용성이 좋은 클래스는 아니다. 〈표 3〉에서 깊이 4가 깊이 5보다 상속도가 높은 이유는 부모 클래스의 수가 3개인 클래스들이 3개 있기 때문이다. 부모 클래스의 수가 1일 때 두 화일의 상속도가 현저히 다른 이유는 Windows 화일은 깊이가 2이상인 클래스가 36개이며, CASE 화일은 4개이기 때문이다.

확장성은 클래스를 쉽게 확장하여 사용할 수 있는 정도를 말한다. 값은 최대 1값과 최소 0값을 기준으로 하였다(3.2.5절 참조). 〈표 3〉에서 Windows 화일의 확장성이 CASE 화일보다 낮았다.

〈표 3〉 객체지향언어면의 품질 메트릭 비교
〈Table 3〉 Comparision of quality metrics in objected oriented language

	메트릭의 종류		화 일 명	
			Windows	CASE
상속도 ()안은 클래스 수를 의미	클래스 평균 상속도		3.099	1.906
	평균 부모클래스 수		0.93	0.735
	평균 깊이 수		2.25	1.26
	깊이 수에 따른 상속도	0	0.893(10)	0.99(12)
		1	2.224(12)	2.105(18)
		2	2.798(15)	3.047(4)
		3	3.688(8)	4.759(1)
		4	6.111	
		5	3.985	
	부모 수에 따른 상속도	0	0.84(9)	0.99
		1	3.104(47)	2.418(21)
		2	0(0)	2.025(2)
		3	8.98(3)	
확장성	평균 확장성(EXT)		0.28	0.660
정보 은닉도	평균 정보 은닉도(DEO)		2.346	4.119
	평균 참조자 사용 정도(DUR)		2.688	3.422

〈표 4〉는 품질 메트릭들의 값들에 따른 재사용성을 만족하는 클래스들의 수와 비율을 나타내고 있다. 같은 메트릭도 응용분야에 따라 그 특성이 달라질 수 있으므로 Window 화일과 Case 화일을 그 크기별로 따로 비율을 비교하였다.

〈표 4〉 메트릭 값들에 따른 재사용성의 비교
〈Table 4〉 Comparison of reusability by metric values

	메트릭의 종류	파일명(비율)			
		Windows		CASE	
절차언어면의 품질들	크기	51	88%	25	71%
	노력도	39	67%	26	74%
	순환수 비 위반수	51	88%	22	63%
	문서화	29	50%	33	94%
	절차적 품질들을 모두 만족하는 수	13	22%	15	43%
객체지향언어면의 품질들	상속도	46	79%	34	97%
	정보은닉정도와 참조자사용정도	24	41%	20	57%
	확장성	24	41%	21	60%
	객체지향 품질을 모두 만족하는 수	7	12%	12	34%
절차언어와 객체지향언어의 품질을 모두 만족하는 수		1	1.7%	3	8.5%

절차언어면의 품질에서는 크기를 위반하지 않은 클래수들, 평균 노력도 이하의 클래스들, 순환수 비위반 클래스들, 평균 문서화 정도 이상의 클래스 수들을 구하였다. 절차언어의 품질 메트릭들을 모두 만족하는 클래스들의 수는 각 화일의 크기에 비해 각각 22%, 43%라는 것을 알 수 있다.

객체지향언어면의 품질에서는 깊이 5이하, 부모수가 3이하이면서 상속도가 1이상인 클래스들, 정보은닉과 참조자 사용정도가 1이상인 클래스들, 평균 확장성 이상인 클래스들을 구하였다. 객체지향언어의 품질 메트릭들을 모두 만족하는 클래스 수는 각 화일의 크기에 비해 각각 12%, 34%였다.

절차언어와 객체지향언어의 품질 메트릭들을 모두 만족하는 클래스들의 수는 각 화일의 크기에 비해 각각 1.7%, 8.5%이므로, 모든 품질 메트릭들을 만족하는 후보자 클래스들은 드물다는 것을 알 수 있었다.

실험결과, CASE 화일이 Windows 화일보다 재사용성 메트릭 정도가 높은 이유는 응용분야, 클래스의 크기, 프로그래머의 능력과 최근의 화일일수록 재사용성 정도가 높다는 것을 보여주었다.

5. 결 론

기존의 절차언어 메트릭들은 operand와 operator에 중점을 두었고, 객체지향언어의 품질 측정 메트릭들은 이론에만 치중하여 정량적인 값을 추출하는 연구는 아직 미흡한 단계이었다.

본 논문에서는 KHR 시스템[11]에서 질의 결과 나온 여러 후보자들의 품질 메트릭들을 정량적으로 추출하는 식을 제안하고, 후보자 클래스들의 품질에 따라 가장 적절한 후보자를 선택할 수 있도록 지원하는 평가 지원 환경을 구현하였다.

KHR 시스템의 질의 결과 나온 후보자들은 재사용자가 원하는 기능을 가진 부품명과 유사한 정도값인 weight를 질의로 하여 나온 후보자들이기 때문에 원하는 기능을 가진 여러개의 후보자 클래스들을 선택할 수 있다. 그리고 후보자 클래스들의 품질 측정과 후보자 선택 과정은 소프트웨어 시스템의 생명주기의 코딩 단계, 테스팅 단계와 유지보수 단계에서 참고하여 낮은 품질을 가진 클래스는 수정하여 높은 품질을 가진 클래스를 만드는 데 도움을 줄 수 있다.

그리고 2개의 화일들의 클래스들을 추출하여 제안한 품질 메트릭들의 정의식에 따라 클래스들은 분석하여 실험하였다. 그 결과, 절차언어와 객체지향언어의 품질 메트릭들을 모두 만족하는 클래스들은 드물지만 재사용자가 원하는 품질 속성에 따라 가장 적절한 후보자를 정량적으로 추출할 수 있었다.

앞으로의 연구방향은 재사용가능한 클래스들의 품질과 정보를 자동으로 분석하고 평가하는 도구와 원시코드를 분석하여 도큐먼트와 설계단계의 명세서를 발생시키는 역공학에 관한 연구가 필요하다.

참 고 문 헌

[1] Motoei Azuma, "The Model Metrics for Software Quality Evaluation Report of the japanese national working group," IEEE Software, pp. 64, 1990.

[2] I.Jackson, "OO Software Engineering," Addison Wesley, 1992.

[3] Chidamber, S.R and Kemerer, C.F, "Towards Metrics for Object Oriented Design," OOPSLA '91, pp.191-211, Oct. 1991.

[4] R.V.Binder, "Design for Testability in OOS," Comm of the ACM, Vol.37, No.9, Sep. 1994.

[5] B.A.Burton, R.W.Aragon and L.A.Mayers, "The reusable S/W engineering," IEEE Software, Jul. 1987.

[6] J.M.Biemand and J.X.Zhao, "Reuse Through Inheritance:A Quantitative study of C++ Software," SSR.'95, IEEE, pp.28-30, Apr. 1995.

[7] R.S.Arnold and W.B.Frakes, "Software Reuse and Reengineering," IEEE Software, pp.476-483, 1991.

[8] R.Prieto and P.Freeman, "Classifying Software for Reusability," IEEE Software, Vol.4, No.1, pp. 6-16, Jan. 1987.

[9] JunAng UNIV Software lab., "Software reuse system by object oriented technique," CARS 1.0 Mannual

[10] G.J.Myers, "Composite/Structured Design," van Nostrand Reinhold, 1978.

[11] 김재생, 송영재, "Environment of the retival system for C++ reusable components," Proceeding of the IASTED International Conference, Jul. 1995.

[12] A.Porter, "C++ Programming for Windows," McGraw Hill, 1993.

[13] D.E.Brubaugh, "Object-Oriented Development," John Wiley & Sons, Inc., 1994.

[14] G.Caldiera and R.Basili, "Identifying and Qualifying reusable Software components," IEEE Software, Feb. 1991.

[15] S.Henry and R.Goff, "Complexity Measurement of a graphical P/L," IEEE Software, Nov. 1989.

[16] E.H.Khan, M.Al-A'ali and M.R.Girgis, "OOP for Structured Procedural Programmers," IEEE Software, Oct. 1995.

[17] 김형주, "객체지향시스템," 동아출판사, pp.229, 1993.
[18] G.A.Miller, "The Magical Number Seven Plus or Minus Two:Some Limits on our Capacity to Process Information," Psycological Reviews, Vol. 63, pp.81-86, 1956.
[19] 송영재, "C 언어로 구현한 소프트웨어 공학," 홍능과학출판사, 1991.
[20] J.Rumbaugh and W.Lorensen, "Object Oriented Modeling and Design," Prentice Hall, 1991.

김 재 생

1988년 경희대학교 전자계산공학과(학사)
1990년 경희대학교 전자계산공학과(석사)
1995년 경희대학교 전자계산공학과 박사과정 수료
1993년~현재 경희대학교 전자계산공학과 시간강사
관심분야:소프트웨어공학, OOP, S/W 재사용

송 영 재

1969년 인하대학교 전기공학과(공학사)
1976년 일본 Keio University 전산학과(공학석사)
1979년 명지대학교 대학원졸(공학박사)
1971년~1973년 일본 Toyo Seiko 연구원
1982년~1983년 미국 Univ. of Maryland 전산학과 연구교수
1985년~1989년 IEEE Computer Society 한국지회 부회장
1984년~1989년 경희대학교 전자계산소장
1976년~현재 경희대학교 전자계산공학과 교수
1993년~1995년 경희대학교 교무처장
1996년~현재 경희대학교 공과대학장
관심분야:소프트웨어공학, OOP/S, CASE 도구, S/W 개발도구론, S/W 재사용