한국산학기술학회논문지
Vol. 4, No. 4, pp. 346-351, 2003

Java프로그램에 대한 품질 및 복잡도 메트릭스 평가 시스템 구현

이상범* · 김경환**

Development of A System for Quality Assessment and Complexity Metrics of Java programs

Sangbum Lee* and Kyonghwan Kim**

요 약 소프트웨어의 규모가 커지고 복잡해지고 있음에도 불구하고 한편으로는 개발기간의 단축, 코스트 절감, 생산성, 품질 향상 등이 요구되어지고 있다. 질 좋고 효율적인 소프트웨어를 구현하기 위해 예전부터 수많은 기법과 방법들이 제안되었고 구현되었다. 대표적인 것으로 다양한 CASE 도구, 프로세스 평가모델(CMM, SPICE, ISO9000), 메트릭스 등이 있다. 하지만 이러한 개발 지원 방법들은 개발자 각 개인의 생산성을 위해 지원하기 보다는 주로 프로젝트 전체 관리를 위해서 사용되어지고 있는 것이 일반적이다. 따라서 이러한 기법과 방법들을 개별 프로그래머의 개발과정에 사용하는 것은 부적절하다. 본 연구에서는 객체지향적 소프트웨어 개발방법론을 적용한 개발 프로세스를 개발조직의 평가개선보다는 개발자 개인의 작업향상과 품질향상에 위한 기법을 개발하였다. 특히 개발프로세스 중 코딩 단계에서 생산되는 생산물(source code)의 품질평가를 할 수 있는 평가 시스템을 제작하였다. 구체적으로 이 평가 시스템은 Java 프로그램에 대한 품질평가로서 단순히 메트릭스 값만을 보여 주는 것이 아니라, 개발자가 작성한 소스코드, 클래스(class)의 계층구조를 동시에 보여준다.

Abatract In spite of the size and complexity of software becomes large and complicated, the demand of rapid development, cost reduction, good productivity and good quality software is increasing in these days. Many methods were proposed for efficient software development such as various Case tools, Metrics, Process improvement model (CMM, SPICE, ISO9000) and etc. However, most of them are useful to manage the whole projects rather than an individual programming. In this paper, we introduced a system for quality assessment and complexity metrics for Java programs to assess the individual programmer's quality rather than team's quality. This system shows not only the metrics value for quality assessment but also the source code and the structure of classes simultaneously.

Key Words : Software Metrics, Complexity, Object-Oriented, Process

1. 서 론

최근 소프트웨어의 응용범위가 확대되어지면서 소프트웨어의 규모가 대형화되고 복잡도가 높아짐에도 불구하고 개발기간의 단축, 개발경비 절감, 품질 및 생산성 향상이 더 어느 때보다 크게 요구되어지고 있다. 질 좋고 효율적인 소프트웨어를 개발하기 위한 연구의 결과로 많은 기법과 방법이 개발되었고 이러한 것들을 통해 많은 효과를 가져다 왔다고 할 수 있다. 대표적인 것으로 컴포넌트를 이용한 소프트웨어 재사용, CASE (Computer Aided Software Engineering)를 이용한 소프트웨어 자동화 개발도구, 그리고 품질을 평가하는 메트릭스, 효율적인 프로세스 평가 모델 등이 있다.

90년대 이후부터는 객체지향개발방법론이 주목을 받으면서 이를 지원하는 언어와 개발환경이 많이 개발되어 기존의 구조적기법에 비해 높은 생산성과 유지보수의 우수성을 보여주고 있다. 객체지향언어는 부품의 재사용성(reuse)을 효율적으로 구현할 수 있다는 장점이 있어 컴포넌트를 이용한 재사용으로 인해 많은 분야의 소프트웨어개발에 적용되고 있다. 재사용성이란 개발된 모듈, 문서 등을 요구변경에 따라 일부 수정해서 다시 사용할 수 있는 개념으로 적은 비용으로 생산성향상 및 품질향상이 실현 될 수 있다. 일본에서는 소프트웨어의 재사용성으로 년간 14%의 생산성향상을 얻었다고 보고하였다[1].

*단국대학교 전자컴퓨터학부, 부교수
sblee@dankook.ac.kr
**일본 SRA 소프트웨어 개발 연구소, 연구원
k-kim@sra.co.jp
이 연구는 2003학년도 단국대학교 대학연구비의 지원으로 연구되었음

최근에는 소프트웨어 개발 과정을 연구하여 비효율적인 부분을 없애고 천편일률적인 절차를 벗어나 소프트웨어 성격이나 각 단계별의 프로세스를 개선하는 것으로 생산성과 품질, 코스트 절감을 할 수 있다는 것이 입증되어 이에 대한 연구가 많이 진행되고 있다. CMM(Capability Maturity Model)[2,3,4,5], SPICE(Software Process Improvement and Capability determination)[6,7] 및 ISO 9000시리즈[6] 등은 개발프로세스를 평가하고 개선하기 위한 모델로서 알려져 있다. 개발프로세스의 개선은 현재 개발프로세스의 상태를 파악·분석하고, 분석결과에 기반을 둔 개선책의 작성과 실행으로 나누어서 실시된다. 개발프로세스의 상황·분석을 보다 객관적으로 실시하기 위해서는 신뢰성 높은 데이터 및 메트릭스를 사용한 분석이 필요하다. 하지만 이들 수법은 개발자 개인의 작업을 평가하고 개선하기보다는 개발프로젝트 전체에 대해서 원활하게 진행할 수 있도록 하는데 목적을 두고 있다.

따라서 본 연구에서는 개발조직 전체의 평가·개선보다는 개발자 각 개인의 작업향상과 품질향상에 초점을 두었다. 특히 본 논문에서는 주로 개발프로세스에 있어서 코딩 단계에서 생산되는 생산물(product)의 품질평가를 지원하는 것을 목적으로 개발한 평가 시스템에 관해서 기술한다. 이 시스템은 Java 프로그램에 대한 품질평가로서 단순히 메트릭스 값만을 보여 주는 것이 아니라, 개발자가 작성한 소스코드, 클래스(class)의 계층구조를 동시에 보여줄 수 있도록 하였으며, 또한 기준치 보다 큰 메트릭스 값은 "**"로 표시하도록 하였다. 이처럼 개발자가 복잡도 메트릭스를 사용해서 프로그램의 복잡한 부분을 특정하고, 그 부분에 대해서 보다 신중히 테스트를 실시할 수 있도록 알려주며 또한 장래에 재사용성을 고려 했을때 프로그램의 설계구조를 수정해야할지 어떨지를 판단가능 하도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구의 기반이 되는 객체지향 메트릭스 및 기존의 품질평가 시스템과 PSP를 소개한다. 3장에서는 시스템의 설계 및 구조를 설명하고, 4장에서는 시스템을 사용한 평가실시 및 그 결과 데이타를 분석하였다. 5장에서는 결론 및 향후연구 방향을 제시하였다.

2. 관련연구

프로세스 개선에 있어서 메트릭스는 주로 프로젝트 관리를 위해서 이용되어지고 있고, 개발자에 대한 지원으로서 이용되어지는 일은 많지 않았다[8]. 개발자가 복잡도 메트릭스를 사용하는 것으로 프로그램 중 특히 복잡하게 되어있는 부분을 특정할 수 있고 프로그램의 구조설계 등의 수정을 행할지 어떨지의 판단 및 테스트를 집중적으로 행할 부분의 특정(特定)이 가능하게 된다. 특히 개발이 개인이 아니라 팀으로서 행하여지는 경우라든가 다른 개발자가 작성한 코드를 이어받아서 개발할 경우 등 개발자가 내부를 상세히 알고 있지 않을 경우에는 보다 객관적으로 평가할 수 것은 중요한 이점이 된다. 하지만 위에서 기술한 것처럼 지금까지 대부분 관리를 위해서 이용되어져 왔고, 개발자 각 개인의 품질 향상에는 등한시 해왔다. 다음은 기존의 메트릭스 시스템 및 개발자 개인의 프로세스 개선에 대해서 간단히 소개한다.

2.1 기존의 메트릭스 시스템과 PSP

1) QUALMS(Quality Analysis and Metrication Software)[9]

소프트웨어에 관한 면밀한 분석과 품질 평가하는 도구로서, 구조 메트릭스나 복잡도 메 트릭스는 prime flowgraph에 바탕을 두고 있다. QUALMS가 측정하는 메트릭스는 다음과 같다.

- 단순한 구조 특성 측정 메트릭스: prime flow-graph의 길이, 가장 큰 prime 중첩의 길이, 경로수
- 결합한 구조 특성 측정 메트릭스: McCabe, Basili

2) ESCORT[9]

C/C++언어로 작성된 원시프로그램에 정적분석을 행하며, 원시프로그램을 수정하고 평가 하여 소프트웨어의 품질향상을 도모한다. ESCORT는 소프트웨어 부품의 보수성 관점에 기인, 원시 프로그램을 측정하는 대상에 관해서 5가지의 특징을 지나고 있다. ①보수성 있는 소프트웨어 부품의 품질 향상도모 ②기계적, 자동적으로 측정 ③객관적으로 평가 ④처리결과에 따른 품질개선 방안을 제시 ⑤다양한 모듈과 부품 작성체제를 정립

3) PSP(Personal Software Process)[10]

CMM, ISO9000, SPICE가 조직의 프로세스에 초점을 두고 있는 것에 비해 PSP는 소프 트웨어 개발자 개인의 훈련, 품질개선, 공수(工數)예측 등에 초점을 맞추고 있다. SPICE, ISO9000, CMM은 소규모의 소프트웨어 개발조직에 적용하는 것이 어렵다는 것과 좋은 작업을 촉진하지만 그 작업이 정말로 좋은지를 보증하지 않는다. 그러나 PSP는 개발자 자신이 프로그램을 작성할 때의 프로세스와 그 성과를 알고, 개선을 위해 개인목표를 설정, 작업을 측정·분석하고 목표에 적합시키도록 프로세스를 수정할 수 있게 된다. 이렇게 해서 개발자는 자신의 퍼포먼스를 예측하거나 품질을 관리할 수 있는 능력을 얻게 된다. PSP는 7개의 스텝으로 이루어져 있다(그림 1). 각 스텝에는 그 스텝에서의 프로세스를 가이드하는 스크립트, 데이터수집에 이용하는 폼 및 템

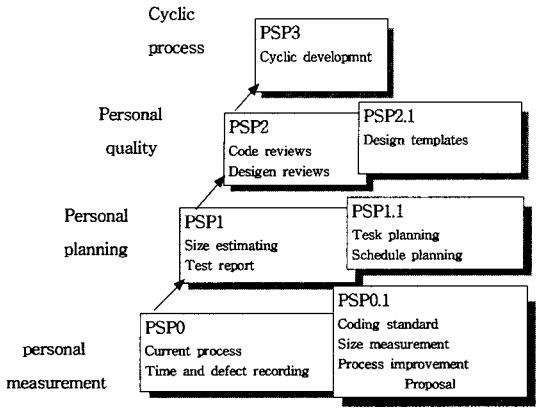

그림 1. PSP 프로세스 진화 과정

플렛이 준비되어있다.

2.2 객체지향 메트릭스

소프트웨어 메트릭스는 소프트웨어 생산물 및 소프트웨어개발·보수의 프로세스에 관계하는 정량적 척도이다. 예를 들어 사양서로부터 소프트웨어 기능을 축출해서 개발 코스트를 예측하는 FP(Function Point)가 있으며, 비객체지향에 대한 복잡도 메트릭스로서 McCabe의 순환수(cyclomatics number), Halstead 메트릭스가 있다. 또한 객체지향으로 개발된 대표적인 메트릭스로서 Chidamber와 Kemerer(이하 C&K메트릭스) [11,12]가 있다. C&K 메트릭스는 6종류의 복잡도 메트릭스를 제안하고 있으며 본 고에서 제안하고 있는 메트릭스 평가 시스템도 C&K의 메트릭스 기준에 따라 자동적으로 평가하고 있다(표1).

3. 평가 시스템 설계 및 구조

3.1 시스템의 개요

본 시스템은 복잡도 메트릭스를 사용해서 소스 코드의 품질평가를 정량적으로 행하기 위한 시스템이다. 복잡도 메트릭스를 계측해서 소스 코드 안에서 구조적 결점을 개발자에게 통지할 수 있는 기능을 갖도록 하였다.

표 1. 객체지향 평가 메트릭스

	평가 메트릭스	내용
1	**WMC** 클래스당 가중치를 갖는 메소드 수 :Weighted Methods per Class	계측대상 클래스C1이 메소드(method) $M_1, \ldots M_n$을 가지고 있다라고 간주한다. 그리고 이들 메소드의 복잡도를 각각 $c_1, \ldots c_n$으로한다면 $WMC = \sum_{i=1}^{n} ci$ 이다. 만약 모든 메소드의 복잡도가 동일하다라고 가정하면 WMC을 메소드 수로서 한다. 본 연구에서도 이 가정을 사용한다. 이 메트릭스는 클래스의 개발 및 보수에 필요한 노력과 시간을 예측하는데 이용된다.
2	**DIT** 상속 트리의 깊이 : Depth of Inheritance Tree	DIT는 계측대상 클래스의 상속(inheritance)의 깊이를 나타낸다. 다중상속이 허락되는 경우는 최대의 패스 길이를 DIT의 값으로 한다. DIT의 값이 클수록 이해용이성 및 유지보수를 저하시킨다.
3	**NOC** 자식노드의수 : Number of Children	NOC는 계측대상 클래스로부터 직접 종속되어있는 서브클래스의 수이다. 많은 수의 서브클래스는 설계상에 있어서 다른 클래스에게 잠재적인 영향을 줄 가능성이 있다.
4	**CBO** 객체간의결합도 : Coupling Between Object classes	CBO는 어떤 클래스에 관계하고 있는 클래스의 수를 계측한다. 예를들어 2개의 클래스에 있어서 한쪽의 클래스가 다른 클래스의 메소드 및 인스턴스 변수를 참조하고 있을 때, 그 2개의 클래스는 관계하고있다라고 한다. CBO가 클수록 유지보수 및 확장성에 영향 을 준다.
5	**RFC** 클래스의 반응도: Response For a Class	RFC는 어떤 클래스에 관계하고 있는 메세지수를 계측한다. 예를들어 계측대상의 클래스의 메시지와 그들의 메시지로부터 호출되는 메소드의 합으로서 정의된다. 이 메트릭스는 테스트, 디버그의 복잡도에 관계가 있다.
6	**LCOM** 메소드에서 응집도의 결핍성 Lack of Cohesion in Methods	계측대상의 클래스C_1이 $M_1, \ldots M_n$ n개의 메소드를 가지고 있다라고 한다. 그리고 메소드 M_i에의해서 사용되어지는 인스턴스변수의 집합을 I_i라고 한다. $P=\{(I_i, I_j) \mid I_i \cap I_j=\varnothing\}$로 정의하고, $Q=\{(I_i, I_j) \mid I_i \cap I_j \neq \varnothing\}$로 정의한다. 만약 $I_1, \ldots I_n$이 전부 $\varnothing$일 때 $P=\varnothing$로한다. $LCOM=\lvert P\rvert-\lvert Q\rvert$, LCOM이 0보다 작을경우 0로 한다. LCOM이 크다라는 것은 변수를 공유하고 있는 부분이 많다라는 의미로 유지보수를 어렵게 만든다는 것을 시사한다.

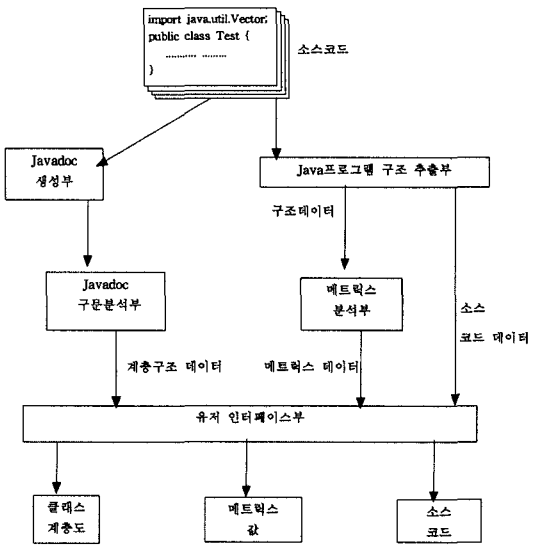

그림 2. 시스템 구성도

자바 소스코드로부터 C&K 메트릭스를 계측하고 메트릭스에 바탕을 둔 분석을 행하기 위한 시스템이다. 시스템은 자바로 구현하였으며 시스템의 구성도는 그림 2와 같다.

위 시스템의 Javadoc생성부는 소스파일 내에서 선언한 클래스, 인터페이스, 생성자, 메소드, 필드를 기술한 HTML문서를 생성한다. 그리고 이 문서를 Javadoc 구문 분석부에서 HTML문서를 파싱해서 클래스의 계층구조만을 추출해낸다. Java프로그램 구조 추출부는 Java소스 프로그램의 문법을 분석하고 정의된 클래스에 대해서 클래스 변수, 인스턴스 변수, 메소드 등을 식별한다. 또한 각 메소드 정의 내부에서 변수, 함수(다른 클래스의 메소드)를참조하고 있는지를 분석한다. 그 분석결과를 구조 데이터로 한다. 메트릭스 분석부는 구조 데이터를 바탕으로 C&K 메트릭스를 계산한다. 계층구조 데이터, 메트릭스 데이터, 소스코드 데이터는 GUI부에 전달되고 사용자의 요구에 따라서 계층도, 메트릭스 값, 소스코드가 표시된다.

3.2 시스템의 기능

위 그림 3에서 알 수 있는 것처럼 본 시스템은 클래스 계층도, C&K 메트릭스에 따른 복잡도 평가, 소스코드 표시부 3가지의 정보를 GUI에 의해서 표시하는 기능을 가지고 있다.

(1) 클래스 계층도

그림 4는 소스파일로부터 클래스의 계층구조, 생성자,

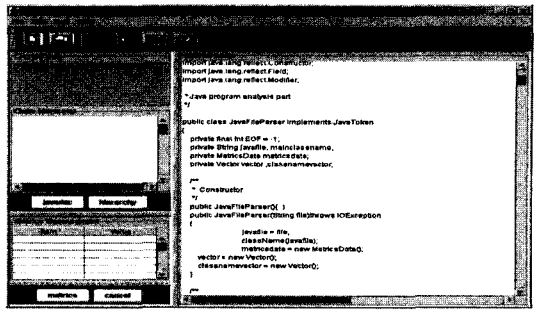

그림 3. 평가 시스템의 구성 화면

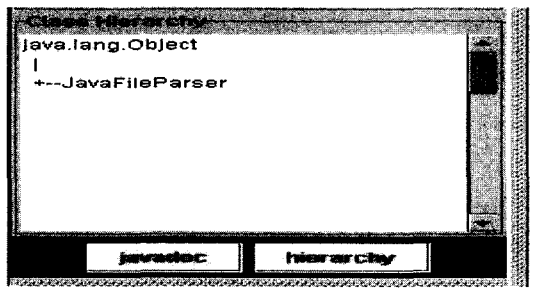

그림 4. 클래스의 계층표시

메소드 등 문서화를 지원하는 JavaDoc을 사용해서 HTML문서를 생성하고, 생성된 HTML 문서를 구문분석해서 계층 구조를 기술한 부분만을 추출해낸다. 그리고 이 계층도를 Class Hierarchy란에 출력한다.

(2) C&K메트릭스에 따른 복잡도 평가

소스코드에 대한 C&K 메트릭스의 값과 그이외의 기본메트릭스(NOA, NCA, NIV)도 계 산하였다. 그리고 기준치보다 측정값이 큰 경우 복잡하다라는 것을 보여주기 위해 측정 값과 함께 "**"이 출력되도록 하였다. 예를 들어 WMC의 기준치를 20이라고 가정하고 계 측대상 클래스의 WMC가 25라고 하는 값이 측정되었을 때 기준치 보다 크다 라는 것을 알려주기 위해 측정값 옆에 "**"가 자동적으로 붙여진다(그림 5).

(3) 소스코드

기존의 메트릭스 측정 시스템과 같이 단순히 측정값만 보여주는 것이 아니라 소스 코드로부터 메트릭스 값이 어떻게 얻어지고 있는지를 보여주기 위해 소스코드도 보 여주도록 하였다(그림 3 참조).

3.4 기준값 설정

기준치보다 측정값이 큰 경우 복잡하다라는 것을 의미한다. 즉, 메트릭스값이 큰 클래스를 발견하면, 그것이 복잡한 클래스가 된다. 또한 클래스종류(GUI)에 따

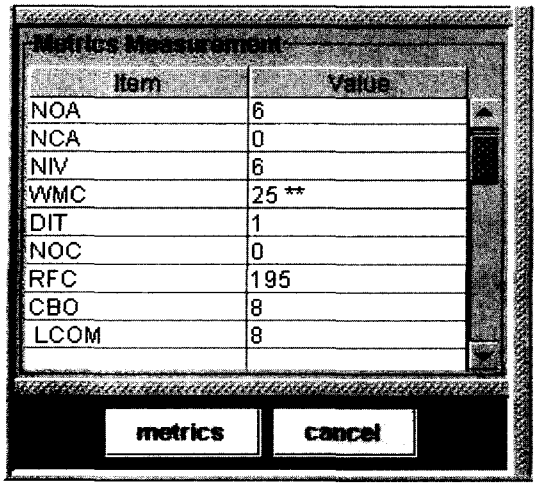

그림 5. 메트릭스 표시창

라서 메트릭스 값의 유효성의 차이가 있기 때문에 우리는 클래스의 종류마다 기준값을 설정하고 계측대상의 클래스 측정값과 기준값과의 차이로부터 그 클래스가 복잡한지 어떤지를 판정하였다. 한편 메트릭스 기준치(Threshold)를 설정하기 위해 과거에 수집된(컴퍼일러 프로그램) 클래스의 메트릭스 값에서 평균값을 구했고(표 2), 특히 GUI부분(JDialog/JFrame)의 메트릭스 기준값은 인터넷상에서 수집한 4개의 프로그램의 평균값을 이용하였다.

4. 평가실험 및 분석

본 시스템을 2002년도에 모 대학교 전자컴퓨터 학부생들이 작성한 프로그램을 실험 데이터를 적용하였는데, 미니 컴파일러로서 GUI를 사용 데이터 입·출력이 이루어지는 프로젝트이다. 프로그래밍언어로서는 자바를 사용하고 있다.

4.1 실험데이터

개발된 3명의 프로그램, 19개의 클래스에 대해서 앞에서 제시한 품질평가 기준에 대해 실제 미니컴파일러 샘플 프로그램을 적용해 보았다. 그 중 Scanner 부분에 대한 메트릭스 값을 보인다.

참고 : WMC 메소드 수
DIT 클래스 상속의 깊이
RFC 클래스 반응의 수
CBO 클래스간의 결합수
LCOM 메소드의 응집도
NOC 계측대상의 클래스로부터 직접도출되는 서브 클래스의 수

4.2 클래스의 분류(GUI)

본 데이터에서 학생들은 GUI로서 크게 JFrame, JDialog, JTable등을 사용하고 있다. 앞에서 언급한 것처럼 클래스의 종류(자바에서 Swing)에 따라 메트릭스 값의 유효성에 차이가 있기 때문에 본 논문에서도 클래스마다 분류하였다.

1) JFrame
여러 개의 컴포넌트를 가지는 경우, 이들을 관리하기 위한 코드가 JFrame으로부터 이어받은 클래스에 기술한다(표 4).

2) JDialog
사용자(User)로부터 입력받는 부분과 사용자에 대해서 에러 메시지를 출력하는 부분이 주 로 기술된다 (표 4).

4.3 분석

실험결과 학생 3명 모두 DIT, NOC의 값이 기준치보다 작은 값이 추출되었지만, CBO의 메트릭스에서 학생 2, 3이 기준치 이상의 값이 추출되었다. 또한 LCOM

표 2. 메트릭스 기준치(Scanner부분)

C&K 메트릭스	WMC	DIT	NOC	RFC	CBO	LCOM
기준치	12	2	1	87	6	6

표 3. 실험 결과 데이터

학생 \ metrics	WMC	DIT	RFC	CBO	LCOM	NOC
S1	8	1	91	5	8	0
S2	11	1	103	7	11	0
S3	20	2	110	8	10	1

표 4. JFrame/ JDialog의 메트릭스

평가 / 항목	WMC	DIT	NOC	RFC	CBO	LCOM
JFrame 기준값	9	6	0	41	9	4
측정값	12	6	0	49	11	8
JDialog 기준값	7	6	0	56	7	5
측정값	3	6	0	41	9	0

메트릭스는 3명 모두가 기준치를 초과하였다. 이 데이터의 결과를 참조로 분석해 보면 특히 학생2, 3에게는 모듈의 응집력을 높이는 방법 및 클래스간의 의존성을 줄일 수 있는 방법에 대해서 학습 시켜줄 필요가 있다고 생각된다. 또한 학생1과 학생2의 NOC 메트릭스의 값을 보면 2명의 학생 모두 0값이 추출되었다. 상속(Inheritance)은 객체지향언어에 있어서 재사용을 높이는 중요한 기능중 하나이다. 실제 소스 코드를 분석 해본 결과 중복되어지는 부분이 있었는데 여기에 클래스의 메소드와 인수턴스 변수를 재사용할 수 있는 상속을 사용했더라면 중복되는 코드 없이 간단히 프로그램을 할 수 있었다. 이것을 볼 때 이 2명의 학생은 상속에 대한 개념을 모르고 있거나 확실하게 이해하고 있지 않다고 보여진다. 따라서 이들 학생에게는 상속에 대한 교육이 필요하다고 생각되어진다. 한편 GUI부분에서 특히 JDialog는 인터넷상에서 수집한 데이터의 WMC 기준치보다 작은 값이 측정되었다. 이것은 다이얼로그에 많은 기능을 갖지 않도록 하였기 때문이다. 실제로 소스 코드를 보더라도 사용자로부터 입력받는 간단한 기능을 가지고 있었다. 이 결과를 볼 때 학생들은 다이얼로그의 역할을 잘 이해하고 있다고 생각되어진다.

5. 결론 및 연구방향

본 연구에서는 개발조직 전체의 평가·개선보다는 개발자 각 개인의 작업향상과 품질향상에 초점을 맞추었다. 특히 개발프로세스에 있어서 코딩 단계에서 생산되는 원시코드와 테스트를 지원하는 것을 목적으로 시스템에 관해서 기술하였고, 실제로 간단한 미니 컴퍼일러에 대해서 평가를 실시하였다. 이후 과제로서 ①좀 더 큰 프로젝트 그리고 보다 많은 프로젝트에 대해서 메트릭스를 수집, 시스템의 유효성을 평가할 필요가 있다. ②본 시스템과 PSP를 사용해서 소스코드뿐만이 아니라 개인의 모든 프로세스를 관리 할 수 있도록 시스템을 확장한다. ③C&K메트릭스 이외의 메트릭스, 자바이외의 다른 객체지향언어에 적용할 수 있도록 시스템을 확장한다. ④JFrame, JDialog 이외의 GUI컴포넌트에 대해서 평가하고자 한다.

참고문헌

[1] 양해술, 황석형: "CMM모델의 새분화를 통한 능력평가 개선 모델의 설계와 정형화", 한 국정보처리학회 소프트웨어 공학지 제2권 제2호 pp29-39, 1999

[2] 장화식, 박만곤: "소프트웨어 재사용가능성의 정량적 측도", 한국정보처리학회 논문지, 제 2권, 제2호, pp.176-184, 1995.

[3] Alan M.Christie: "Simulation in support of CMM-based process improvement", Journal of System and Software, Vol. 46, pp.107-112, 1999.

[4] Joseph Raynus, 토미노히사시(역): "CMM에 따른 프로세스 개선 입문", 공립출판주식회사, 2002.

[5] CMM, http://www.sra.co.jp/public/doc/GSI/etter/vol.30/CMM/CMM-11.html

[6] 정기원,윤창섭,김태현: "소프트웨어 프로세스와 품질", 홍릉과학출판사,1997

[7] SPICE웹사이트 : www-sqi.cit.gu.edu.au/spice/index.html

[8] 이노우에, 마츠모토,이이다: "소프트웨어프로세스", 공립출판주식회사, 2000.

[9] 최은만, 남윤석: "재사용 소프트웨어 품질평가도구 개발", 한국정보처리학회 논문지, 제4 권, 제8호, pp.1948-1960, 8월, 1997.

[10] W.S.Humphrey: "Introduction to the Personal Software Process", Addison-Wesley, 1995.

[11] S.R.Chidamber and C.F.Kemerer: "A Metrics Suite for Object Oriented Design", IEEE Transaction on Software Engineering, Vol20, No.6 pp.476-493, 1994.

[12] K.Hatano, Y.NOMURA, H.Taniguchi and K.USHIJIMA: "A Method TO Support Refactoring Using C&K Metrics", Proceedings of PYIWIT, pp.112-118, Mar. 2002.

[13] Li.W and HenryS.: "Object Oriented Metrics that predict Maintainability", Journal of Systems and Software,Vol23, No.2 pp. 111-122, 1993.

[14] Roger S. Pressman, 유해영(역): "이론과 실무 중심의 21세기용 소프트웨어 공학- 제 4판", 사이텍미디어, 1997.