Java 프로그램에 대한 복잡도 척도들의 실험적 검증

(An Empirical Validation of Complexity Metrics for Java Programs)

김 재 웅 † 유 철 중 †† 장 옥 배 ††

(Jae-Woong Kim)(Cheol-Jung Yoo)(Ok-Bae Chang)

요 약 본 논문에서는 Java 프로그램의 복잡도를 측정하기 위해 필요한 인자들을 제안하였다. 이러한 인자들을 추출하기 위해 Java 프로그램을 분석하여 객체지향 설계 척도 값들을 계산하고 통계적 분석을 수행하였다. 그 결과 기존의 연구에서 발견되었던 클래스의 크기 인자 외에도 메소드 호출 빈도, 응집도, 자식 클래스의 수, 내부 클래스 및 상속 계층의 깊이가 주요 인자임이 파악되었다. 클래스의 크기 척도로 분류되었던 자식 클래스의 수는 다른 크기 척도들과 다른 성질을 가진다는 것을 발견하였다. 또한 프로그램의 크기가 커지고 결합도가 높아질수록 응집도가 떨어진다는 것을 입증하였다. 그리고 인자 분석을 바탕으로 인간의 인지 능력과 인자의 상관관계를 고려한 가중치를 적용하기 위해 인자별로 회귀 분석을 수행하였다. 보다 적은 척도를 가지고 인자를 설명할 수 있는 회귀식을 도출하였고, 두 그룹에 대한 교차 검증 결과 회귀식이 높은 신뢰도를 가지는 것으로 나타났다. 따라서 본 논문에서 제안한 인자들을 이용하는 경우 Java 프로그램의 복잡도를 측정할 수 있는 새로운 척도로 사용할 수 있다.

Abstract In this paper, we proposed some factors to measure the complexity of Java programs. For the purpose of extracting the factors, we have evaluated the values of object-oriented design metrics with Java programs and performed statistical analyses. As a result, we discovered that there are many factors to measure the complexity of Java program, not only the class size factor on existing research, but also the frequency of method invocations, the degree of their cohesion, the number of children classes, inner class, and depth of inheritance hierarchies. It is also discovered that the number of children classes classified by size metrics has different character corresponding to different size metrics. Moreover, it is proved that as program size big and coupling high, cohesion gets lower. Then, we do regression analysis by each factor in order to apply weight considered human cognitive ability and correlation between factors based on factor analysis results. We derive regression equations to explain factors with fewer metrics. Additionally, the cross-validation for two groups proves that the regression equations driven in our paper are valid throughout the population with a high degree of confidence. Consequently, it is possible to use a new metrics for measuring the complexity of Java programs using these proposed factors in this paper.

1. 서 론

현재의 컴퓨터 환경에서는 하드웨어의 급속한 발전에 비해 소프트웨어 개발 기술이 상대적으로 낙후되고 소프트웨어 크기와 복잡도가 기하급수적으로 증가하기 때문에 소프트웨어의 개발과 유지보수를 더욱 힘들게 하고 있다. 소프트웨어는 하드웨어와는 달리 변형이 가능하며, 논리적 혹은 자체적인 결함에 의해서뿐만 아니라 더 좋은 품질을 유지하기 위해서 유지보수에 많은 시간과 노력을 필요로 하기 때문에 소프트웨어의 유지보수는 시간이 흐를수록 소프트웨어의 개발에 드는 비용의 상당한 부분을 차지하고 있다. 1970년대에 전체 개발비

· 본 연구는 전북대학교 영상·정보 신기술 연구소의 지원을 받았음.

† 정 회 원 : 전북대학교 컴퓨터과학과
jwkim@cs.chonbuk.ac.kr
†† 종신회원 : 전북대학교 컴퓨터과학과 교수
cjyoo@moak.chonbuk.ac.kr
okjang@moak.chonbuk.ac.kr

논문접수 : 1999년 8월 31일
심사완료 : 2000년 11월 1일

중 35~40%를 차지하던 것이, 1990년대에 와서는 70~80%까지 이르고 있어 유지 보수 비용을 관리하는 것이 소프트웨어 공학자들에게 가장 큰 관심사가 되고 있다. 유지보수 비용을 줄이기 위해서는 개발 주기의 초기 단계에서부터 시스템의 품질을 관리할 필요가 있다. 소프트웨어 복잡도 척도는 완전한 소스코드나 설계 표현으로부터 계산된 정량적인 정보를 제공함으로써 유지보수 비용을 감소할 수 있도록 한다.

소프트웨어 개발에서 척도를 사용하기 위해서는 대략 세 단계의 과정을 거쳐야 한다. 첫째, 척도에 대한 프레임워크(framework)를 정의해야 하고, 둘째, 속성과 관련된 적절하고 명확한 자료를 수집하기 위한 과정을 정의하고 확인해야 한다. 마지막으로 그들 속성을 측정하고 해석해야 할 지침을 제시하고, 신뢰성, 유지보수성, 유용성과 같은 다른 외부 속성을 측정할 수 있도록 해야 한다[1].

척도에 대한 프레임워크를 만들기 위해서는 우선 척도의 분류가 필요하다. 그동안 상속 트리나 메소드 호출, 응집도와 결합도 등의 다양한 설계 매개변수를 사용하여 객체지향 설계의 품질을 평가한 다양한 객체지향 척도들이 제안되었다[2, 3, 4, 5, 6]. 그러나 많은 척도들이 속성을 명확하게 분류하는데 실패하여 어떤 척도가 실제로 품질의 중요한 면을 측정하는 지 알 수가 없다. 그러므로 우선 척도들간의 관련성을 규명하는 것이 가장 의미있는 척도를 선택하는데 있어서 중요하다.

설계 척도(design metrics)들은 구현을 시작하기 전에 설계 단계에 적용할 수 있다. 그러나 메소드 호출 등에 관련된 정보는 구현이 완료되기 전에는 불확실하기 때문에 구현이 시작되기 전에 측정을 하면 실제 개발된 산물의 정확한 측정자료를 얻을 수 없게 된다.

본 논문에서는 Java 프로그램의 복잡도를 측정하기 위해 필요한 인자들을 제안하였다. 이 인자들을 추출하기 위해 Briand[7]가 제안한 속성을 만족하는 척도들과 Java 언어의 특징인 내부 클래스(inner class)를 반영한 척도와 크기 척도 등 13개 척도의 관련성을 분석하였다. 인터넷과 JDK 1.2에서 수집한 두 그룹의 Java 프로그램을 분석하여 척도 값들을 계산하고, 통계 분석을 통해 인자들에 대한 회귀식을 도출하여, 도출된 회귀식을 두 그룹에 대해 교차 검증함으로써 타당성을 증명하였다. 본 논문에서 제안한 인자들을 이용하는 경우 Java 프로그램의 복잡도를 측정할 수 있는 새로운 척도로 사용할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 통계 분석 방법과 소프트웨어 척도의 속성을 살펴보고, 3장에서는 측정할 소프트웨어 척도에 대해 정의한다. 4장에서는 자료 분석 환경과 방법론 및 분석 결과를 설명하고, 5장에서는 추출된 회귀식의 교차검증에 의한 평가를 실시하고, 마지막으로 6장에서 결론을 기술한다.

2. 관련 연구

이 장에서는 통계 분석 방법과 Briand[7]가 제시한 소프트웨어 척도가 만족해야 할 속성을 설명한다.

2.1 통계 분석

많은 소프트웨어 척도들이 유사한 구조를 측정하고 있고, 일부 척도들은 매우 높은 상관관계를 가지고 있다. 그러므로 척도들간의 관련성을 규명하는 것이 가장 의미있는 척도를 선택하는데 있어서 중요하다. 인자 분석과 회귀 분석 등의 통계적 분석 방법을 이용하여 소프트웨어 복잡도 척도들간의 관련성을 분석한 몇몇 연구들이 있다[8, 9, 10].

인자 분석은 변수들간의 상관관계를 고려해서 이들 측정치사이에 공유하는 구조를 파악해 내는 기법이다. 이를 통해 연구자들이 주어진 많은 정보에 대해 분석이 용이하도록 적은 수의 인자(factor)로 제시해 주는 분석 방법이다. 최초의 인자 분석 행렬은 변수와 인자간의 관계가 명확하게 나타나지 않으므로 인자를 해석하기 쉽도록 회전된 구성요소를 고려한다. 그 결과로 회전된 구성요소는 변수가 인자에서 무시해도 좋은가 중요한 영향이 있는가를 명확히 보여준다. 그러한 회전을 수행하기 위해 몇 가지 전략이 존재하는데 배리맥스(varimax) 회전 방법을 가장 빈번하게 사용한다. 인자 분석을 수행하기 위한 표본의 수는 50개 이상이어야 하며 일반적으로 측정 변수의 4~5배가 필요하다[11]. 인자 분석에서 얻은 인자 점수를 다중 회귀 분석에서 사용할 수 있다.

Coupal[9]은 Fortran, C, Pascal, PL/I, Modula-2, COBOL 등의 절차지향 프로그램 언어로 작성된 19개의 프로젝트에 대해 32개의 척도를 가지고 인자 분석을 실시하였다. 인자 분석 결과 볼륨을 나타내는 척도가 평균 63%의 변화율을 표현하고, 프로젝트당 평균 4개의 인자가 추출되며 초기 정보의 83.7%를 설명하는 것으로 나타났다. 볼륨을 제외한 다른 인자의 해석은 실행하지 않았다.

상관관계 분석은 측정된 변수들간의 관련성의 정도를 알아보기 위한 것이다. 한 변수의 값이 증가할 때 다른 변수의 값도 같이 커지는 경향을 보일 때 두 변수간에는 양(positive)의 상관관계가 있다고 하고, 반대로 한 변수의 값이 증가할 때 다른 변수의 값이 감소하면 두 변수는 음(negative)의 상관관계가 있다고 한다. 두 변수간의

상관관계의 강도를 나타내 주는 것을 상관계수(correlation coefficient; r)라 하며, $-1 \le r \le 1$ 사이의 값을 갖는다. 상관계수가 0에 가깝게 나타나는 경우는 두 변수간의 관계가 전혀 없는 독립적인 관계임을 의미한다.

다중 회귀 분석(multiple regression analysis)은 하나이상의 독립변수와 종속변수와의 관계를 선형 관계식으로 나타낸다. 변수들과의 상호관계를 분석하고 독립변수의 변화로부터 종속변수의 변화를 예측하기 위해 사용된다. 다중회귀 분석을 수행할 경우 포함된 독립변수들이 높은 상관관계를 가질 경우에는 추정된 계수가 통계적으로 유의하지 않게 나타날 가능성이 높다. 이것은 한 변수의 설명력이 다른 변수에 의해 흡수되는 다중공선성(multicollinearity)이 발생하기 때문이다. 다중공선성의 발생을 방지하기 위해서는 미리 변수들간의 상관관계를 검토하여 상관관계가 높은 두 변수중 하나를 제거하거나, 단계적 변수투입법(stepwise method)을 사용하여 상관관계가 높은 변수들 중 가장 설명력이 있는 독립변수만을 분석에 포함시켜야 한다[11].

Li[10]는 Classic-Ada로 개발된 두 그룹의 소프트웨어 생산품에 대해 회귀 분석을 실시하였다. 클래스당 변화된 라인 수에 의해 유지보수 노력을 측정하고, 10개의 척도와의 관계를 회귀분석으로 측정하였다. 10개의 척도를 가지고 유지보수 노력을 예측한 것과 크기 척도(세미콜론의 수, 어트리뷰트의 수 + 메소드의 수)만을 가지고 예측한 것에는 차이가 있으므로 크기 척도가 유일한 주요 예측자가 아니고 다른 척도들도 유지보수 노력예측에 기여한다는 것을 발견하였다. 분산확대지수(VIF : Variance Inflation Factor)가 50보다 큰 척도를 제거할 때 척도의 수집용이성과 기존의 연구에서 나타났던 몇몇 척도들간의 상관관계에 의해 척도를 소거하였다. 인자의 중요성이나 상관 관계를 반영하지 않고 모든 척도에 대해 회귀 분석을 수행하였다.

2.2 소프트웨어 척도의 속성

최근 몇 년 동안 소프트웨어 척도에 대해 요구되는 속성을 제공하는 논문이 여러 편 발표되었다[12, 13, 14]. 이러한 연구들은 기존에 제안된 척도와 새롭게 제안한 소프트웨어 척도를 검증하기 위해 사용되었다. 그러나 주관적인 기준에 의해 정의된 소프트웨어 복잡도 척도들이 많기 때문에 아직도 그 척도들에 대한 타당성 논의가 계속되고 있다.

품질평가에 필요한 메트릭스를 적용하기 위해서는 일관성있는 개념 정의가 중요하다. 본 연구에서는 코드와 관련있는 크기, 길이, 복잡도, 응집도, 결합도 등의 개념에 대해 Briand[7]가 제안한 속성을 만족하는 척도를 측정 대상으로 하였다. 각각의 개념에 대한 속성은 다음과 같다.

2.2.1 크기(Size)

속성 1 : 시스템의 크기는 항상 0보다 커야한다.

속성 2 : 만일 시스템을 구성하는 요소가 없다면 시스템의 크기는 0이다.

속성 3 : 시스템 S에 m_1과 m_2의 요소가 있고, 두 요소 사이에 공통 부분이 없다면 시스템의 크기는 두 요소의 크기를 합한 것과 같다.

기존에 제안된 크기 척도 중에서 이 속성을 만족하는 척도에는 LOC(Line Of Code), WMC(Weighted Methods per Class), NOC(Number of Children), RFC(Response set For a Classes) 등이 있다[2].

2.2.2 길이(Length)

길이는 두 모듈간 거리의 값을 말하는 것으로 모듈간의 관계성을 의미한다.

속성 1 : 시스템의 길이는 항상 0보다 크다.

속성 2 : 만일 요소를 가지고 있지 않으면 시스템의 길이는 0이다.

속성 3 : 같은 시스템 내에 있는 두 요소들간에 새로운 관계가 생성되어도 새로운 시스템의 길이는 원래 시스템의 길이보다 크지 않다. 새로운 관계는 요소들 사이를 좀더 밀접하게 하기 때문에 요소들간의 길이를 더욱 줄일 수도 있다는 것을 의미한다.

속성 4 : 시스템 S 내에서 분리된 모듈 m_1, m_2간에 새로운 관계가 생성될 때 S의 길이는 감소되지 않는다.

속성 5 : 시스템 S의 길이는 분리모듈 m_1, m_2에서는 m_1과 m_2의 길이의 최대값과 같다.

길이에 관련된 척도는 프로그램의 중첩 깊이(nesting depth)[2], DIT(Depth of Inheritance Tree)[2] 등이 있다.

2.2.3 복잡도(Complexity)

복잡도는 소프트웨어 시스템의 특성과 관련된 메트릭스로서 신뢰성과 유지보수성에 관한 중요한 정보를 예측하는데 사용한다. Briand는 여기에서 심리적으로 영향을 주는 외부 변수를 제외한 소스 코드로부터의 본질적 속성만을 정의하였다.

속성 1 : 복잡도는 항상 0보다 크거나 같다.

속성 2 : 시스템사이의 관계가 없다면 시스템의 복잡도는 0이다.

속성 3 : 시스템의 복잡도는 대칭성을 가진다.

속성 4 : 시스템의 복잡도는 관계가 없는 두 모듈의 복잡

도 합보다 작지 않다.

속성 5 : 두 개의 분리된 모듈로 이루어진 시스템 복잡도는 두 모듈의 복잡도의 합과 같다.

속성 6 : 시스템 요소사이에 관계를 추가하는 것이 시스템의 복잡도를 감소시키지 않는다.

2.2.4 응집도(Cohesion)

응집도는 모듈의 기능적인 응집력을 측정하는 방법으로 프로그램의 다른 부분에서 실행되고 있는 절차들과 상호작용을 요구하지 않으면서 소프트웨어 프로시저에서 단일 작업을 수행하는 정도를 측정하는 것이다[15].

속성 1 : 응집도는 0보다 크거나 같아야 한다.

속성 2 : 모듈내의 메소드사이에 관계가 없다면 응집도는 0이다.

속성 3 : 모듈간에 새로운 관계를 추가해도 응집도는 감소되지 않는다.

속성 4 : 두 개의 관계없는 모듈을 결합하여 생긴 모듈의 응집도는 원래의 두 모듈이 가지고 있던 응집도의 최고값보다 크지 않다.

응집도 속성을 만족하는 척도에는 Coh과 TCC, LCC가 있다[16].

2.2.5 결합도(Coupling)

결합도는 소프트웨어 시스템의 구성요소사이에 상호관련성을 측정하는 방법이다. 소프트웨어 설계시에는 가능한 최저의 결합도를 갖도록 노력하는데, 이것은 모듈들 간의 간단한 연결은 이해하기 쉬운 소프트웨어가 되게 하고, 오류가 한 지점에서 발생해서 시스템을 통해 전파되는 리플 효과(ripple effect)를 줄여주기 때문이다[15].

속성 1 : 결합도는 0보다 크거나 같다.

속성 2 : 호출 메소드가 없으면 결합도는 0이다.

속성 3 : 모듈간에 새로운 관계가 추가되었을 때 모듈들은 서로간에 더 많은 의존관계를 갖게 되므로 결합도는 감소하지 않는다

속성 4 : 두 개의 관련있는 모듈을 병합하여 얻은 모듈의 결합도는 원래의 두 모듈이 가지고 있던 결합도의 합보다 크지 않다.

속성 5 : 두 개의 관련 없는 모듈을 결합하여 생성된 모듈의 결합도는 원래 모듈들의 결합도 합과 같다.

결합도 속성을 만족하는 척도에는 RFC(Response set For a Classes), CBO(Coupling Between Object classes) 등이 있다[2].

3. 소프트웨어 척도 측정

본 논문에서는 Briand[7]가 제시한 속성을 만족하는 객체지향 척도들 중 측정을 자동화하기 용이한 척도들과 Java 언어의 특징을 고려한 척도, 그리고 절차적 패러다임에서 사용되어 온 단순한 크기 척도 등 13개의 척도를 측정하였다. 13개의 척도는 다음과 같다.

WMC(Weighted Methods per Class) 척도[2]는 클래스에 정의된 메소드의 복잡도를 측정하는 것이다. 클래스의 메소드가 가진 제어 흐름이 복잡해질수록 메소드를 이해하기가 더 어려워지므로 메소드를 유지보수하기가 점점 어려워진다. 또한 자식 클래스는 클래스에 정의된 모든 메소드를 상속받으므로 클래스에 정의된 메소드의 수가 많고 복잡할수록 자식 클래스에 미치는 잠재적인 영향이 커진다. WMC는 메소드에 대한 McCabe[17]의 사이클로메틱 복잡도 값의 합으로 계산하였다.

NOC(Number of Children) 척도는 한 클래스가 가지고 있는 직접적인 자식 클래스의 수이다. 한 클래스가 가지고 있는 직접적인 자식 클래스의 수가 많으면 많을수록 상속성 때문에 잠재적으로 영향을 끼칠 수 있는 클래스들이 점점 더 증가하고 유지보수가 점점 더 어려워진다고 할 수 있다. 예를 들어 수퍼클래스에서 정의된 메소드나 변수에 의존하는 클래스가 많은 자식 클래스를 가지고 있다면 이 메소드나 변수의 변화는 자식 클래스에 영향을 끼칠 수 있다.

NOM(Number of Methods)와 NOPM(Number of Public Methods) 척도는 각각 메소드의 수와 공용 메소드의 수이다. 클래스에 정의된 메소드의 수가 많으면 많을수록 클래스의 인터페이스가 더욱 복잡해지고, 자식 클래스에 미치는 잠재적인 영향이 커지므로 유지보수가 어려워진다.

LOC(Lines of Code) 척도는 프로그램 헤더, 선언문, 실행문과 비실행문 모두를 포함하고 주석을 제외한 라인 수를 측정하고 SLOC(Source Lines of Code) 척도는 실행문만을 고려하는 것으로 클래스 안의 세미콜론의 수를 측정한다.

NOM과 NOPM, LOC와 SLOC는 큰 상관관계를 가질 것으로 고려되지만 소프트웨어 척도들의 정확한 관련성을 파악하기 위해 분석에 모두 포함하였다. 중복된 성질을 가지는 척도들은 회귀 분석 과정에서 다중공선성을 배제하기 위해 제거될 것이다.

DIT(Depth of Inheritance Tree) 척도는 상속 트리의 노드로부터 루트까지의 최대 길이이다. 계층에서 클래스의 위치가 깊으면 깊을수록 상속받는 메소드의 수가 증가하여 수행을 예측하기 어렵다. 루트 클래스의 DIT는 0이다. Java 패키지로부터의 상속 계층은 척도

DIT는 0이다. Java 패키지로부터의 상속 계층은 척도 계산에 고려하지 않았다.

NIM(Number of Invoked Methods) 척도는 한 클래스의 응답(response) 집합의 개수를 측정하는 것이다. 한 클래스의 응답 집합은 메소드에 의해 호출되는 모든 메소드들로 구성된다. 한 클래스에 대한 응답 집합이 크면 클수록 그 클래스는 더욱 더 복잡해지기 때문에 유지보수하기가 점점 더 어려울 것이다. NOCB(Number of CollaBorator) 척도는 메소드에서 참조하는 협력자(collaborator)의 수를 측정한다.

LCOM(Lack of COhesion in Methods) 척도는 클래스의 응집도 결핍을 측정하는 것이다. 한 클래스의 응집도는 메소드들이 클래스내의 인스턴스 변수와 얼마나 밀접히 관련되어 있는지에 의해 특징 지워진다. 클래스가 응집력이 크면 클수록 그 클래스를 유지보수하기가 더욱 쉽기 때문에 LCOM 척도가 크면 클수록 그 클래스를 유지보수하기가 점점 더 어려워진다는 것을 알 수 있다. LCOM은 Henderson-Sellers[6]가 정의한 식 (1)을 사용하였다. LCOM은 0에서 2까지의 값을 가진다.

$$LCOM = \frac{\frac{\text{전체 변수 사용 수}}{\text{변수 수}} - \text{메소드수}}{1-\text{메소드수}} \quad (1)$$

LCOM이 Briand[7]의 속성을 만족하지 않지만 Coh가 LCOM의 변형이므로 두 응집도 척도의 관계를 살펴보기 위해 척도 값 측정에 포함하였다.

Coh[15] 척도는 LCOM의 변형으로 클래스의 응집도를 측정하는 척도로 Briand[7]의 속성을 모두 만족하였다. 두 척도 모두 클래스의 메소드에서 사용한 전체 변수 사용 수를 중심으로 응집도를 측정하였다.

$$Coh = \frac{\text{전체 사용 변수 수}}{\text{메소드 수} \times \text{변수 수}} \quad (2)$$

NIC(Number of Inner Class) 척도는 내부 클래스(inner class)의 수를 측정한다. 내부 클래스는 썬 마이크로시스템즈사의 JDK 1.1 이후에 추가된 Java 언어의 특징으로 다른 클래스에 내포되어 클래스의 정의를 할 수 있도록 한 것이다. 프로그램 내부에 포함된 내부 클래스의 수가 많으면 많을수록 메소드 호출이 많이 발생하기 때문에 클래스의 복잡도가 증가될 것이다. DOIC(Depth Of Inner Class) 척도는 내부 클래스의 상속 깊이를 계산한다.

지금까지 본 논문에서 분석할 척도들을 설명하였고, 이를 정리하면 다음 표 1과 같다.

제안된 척도들을 Java 프로그램에 대해 조사하고, Java 언어의 새로운 구조를 측정하기 위해 일부 척도들을 제안하였다. 13개의 척도에 대해 두 그룹에서 수집된 데이타로 인자 분석을 수행하여 많은 척도들이 같은 성질을 측정하고 있다는 점을 밝히고, 같은 인자에 속하는 척도간에는 높은 상관관계를 가지므로, 이 중 일부 척도만으로도 인자의 설명이 가능하다는 것을 회귀 분석을 통해 입증하고자 한다.

표 1 측정 소프트웨어 척도

척 도	설 명	척도 분류
WMC	클래스내 메소드의 사이클로메틱 복잡도(Cyclomatic Complexity)값의 합	크기(Size)
NOC	자식 클래스의 수	〃
NOM	클래스에 정의된 메소드(Method)의 수	〃
NOPM	클래스에 정의된 공용 메소드(Public Method)의 수	〃
LOC	실행문과 비실행문 모두를 포함하고 주석을 제외한 라인 수	〃
SLOC	실행문만을 고려한 라인 수	길이(Length)
DIT	클래스 상속 깊이	결합도(Coupling)
NIM	호출된 메소드(Invoked Method)의 수	〃
NOCB	메소드에서 참조하는 협력자(Collaborator)의 수	응집도(Cohesion)
LCOM	클래스의 응집도의 결핍	〃
Coh	클래스의 응집도	Java 특성
NIC	내부 클래스(Inner Class)의 수	〃
DOIC	내부 클래스(Inner Class)의 상속 깊이	

4. 데이타 분석

4.1 데이타 수집과 분석 환경

정형화된 프로그램과 실제 업계에서 개발되어 사용되고 있는 프로그램과는 구현된 소스 코드를 분석하였을 때 구성 성분에서 많은 차이가 나기 때문에, 그 차이가 복잡도를 측정하는데 어떤 영향을 주는 가를 파악하기 위해 3장에서 살펴본 척도들을 두 그룹의 프로그램에 각각 적용하였다. 첫 번째 그룹은 객체지향 원칙을 충실히 준수하여 코드 자체가 공학화되었다고 할 수 있는 썬 마이크로시스템즈사의 JDK 1.2를 구성하는 awt 패키지에서 220개의 프로그램(AWT 그룹)을 사용하였고, 두 번째 그룹은 사용자 인터페이스와 관련되어 실제 업계에서 구현되어 사용중인 177개의 Java 프로그램(UIS 그룹)을 사용하여 척도 값을 추출하였다. 두 그룹 모두 500 SLOC 이하의 프로그램을 대상으로 하였다. 척도 값의 측정은 Banda의 Java Source Metrics[18]와 Andrew와 Rajesh의 JMetric[19] 등의 자동화 도구를 주로 사용하였고, 일부 프로그램은 C++로 구현하였다.

397개의 Java 프로그램에 대해 수집된 데이타를 분석하기 위해 기술 통계 분석(descriptive analysis), 상관관계 분석(correlation analysis), 인자 분석(factor

analysis), 회귀 분석(regression analysis)을 사용하였다. 실제 데이타의 일부를 부록에 첨부하였다.

4.2 분석 결과

이 절에서는 두 그룹에 대해 추출된 13개의 척도 값을 가지고 다양한 통계 분석을 수행하여 척도들간의 관계성을 파악하고, 그 결과가 갖는 의미에 대하여 기술한다.

4.2.1 기술 통계

데이타 집합의 특징을 조사하기 위해 두 그룹에 대한 13개의 척도 값을 분석하였다. AWT 그룹의 프로그램을 분석한 결과 표준 편차가 평균값보다 큰 척도들이 상당히 많았지만 인자 분석에 적합한 표본인가를 검증해주는 Kasier-Meyer-Olkin의 KMO 통계량을 구한 결과 0.787로 인자분석에 적합한 표본으로 판단되었다.

메소드 호출과 관련 있는 NIM 측정값이 다른 언어에 비해 상당히 많은데 이것은 객체지향 원칙에 충실하였을 경우에 메소드 호출이 많아진다는 것과 관련이 있다[2]. 상속과 관련된 DIT 척도와 NOC 척도 값이 상당히 낮게 나왔다. 이것은 대부분의 클래스들이 일반적으로 매우 적은 수의 자식 클래스를 가지고 단지 매우 적은 수의 클래스들만이 많은 자식 클래스를 가진다는 것을 보여준다. 응집도와 관련해서 클래스내의 전체 변수 사용 수를 계산하는 척도 LCOM과 Coh를 분석한 결과 대부분의 프로그램들이 낮은 응집도를 가진다는 것을 보여준다. Java의 특성인 내부 클래스도 거의 사용하지 않는 것으로 나타났다. UIS 그룹도 AWT 그룹과 비슷한 측정값을 나타냈다. 메소드의 수는 별 차이가 없었으나 메소드 호출과 라인 수는 상당한 차이가 있었다. 결합도 척도인 NIM과 NOCB 값이 높으면서 응집도 척도인 LCOM과 Coh 값도 높은 형태를 보여주었다. 사용자 인터페이스와 관련 있는 프로그램이어서 결합도 척도 값이 높게 나타났고, 또한 내부 클래스의 사용도 AWT 그룹보다 많았다. DIT와 NOC에 대해 낮은 평균값을 가지는 것은 상속이 시스템 내에서 드물게 사용되었다는 것을 보여준다. 다른 연구[2]에서도 유사한 결과를 보이고 있다. 두 그룹을 LOC와 SLOC 척도 값을 기준으로 상대 비교하였을 때 AWT 그룹에서는 WMC, NOM, NOPM 척도 값이, UIS 그룹에서는 NIM 척도 값이 상대적으로 큰 것을 알 수 있었다.

4.2.2 상관관계 분석

표 2와 3의 상관관계 행렬로부터 두 그룹에서 측정한 척도 값들의 상관관계를 볼 수 있다.

많은 척도사이에 높은 상관관계가 있는데 특히 크기 척도와 결합도 척도들이 높은 양(positive)의 상관관계를 가지는 것을 볼 수 있다. 응집도 척도인 Coh는 두 그룹에서 모두 다른 척도와 음(negative)의 상관관계를 가지는 것으로 나타났다. 특히 크기 척도와 결합도 척도에 대해 AWT 그룹에서 0.222~0.309의 음의 상관관계를 가지고, UIS 그룹에서 0.218~0.478의 높은 음의 상관관계를 가졌다. 이것은 프로그램의 크기가 커지고 결합도가 높아질수록 응집도가 떨어진다는 것을 나타낸다.

두 그룹의 분석에서 NOM, NOPM과 NIC, DOIC, NOCB와 DIT, NIC와 DIT사이에서 상관계수의 부호가 다른 결과가 나타나고 있는데, 이것은 UIS 그룹에 포함되어 있는 프로그램들이 AWT 그룹의 프로그램에 비해 높은 결합도 척도 값을 가지고 내부 클래스의 사용이 많다는 프로그램 성질에 기인한다. Coh는 응집도를, LCOM은 응집도의 결핍을 측정하는 척도이기 때문에 -1에 가까운 측정값을 보여야 하지만, 척도 값의 범위

표 2 AWT 그룹에 대한 상관관계

	WMC	NOC	NOM	NOPM	LOC	SLOC	DIT	NIM	NOCB	LCOM	Coh	NIC	DOIC
WMC	1												
NOC	0.045	1											
NOM	0.789	0.084	1										
NOPM	0.727	0.068	0.956	1									
LOC	0.953	0.038	0.734	0.675	1								
SLOC	0.957	0.030	0.709	0.653	0.988	1							
DIT	0.004	-0.078	0.000	0.002	-0.038	-0.034	1						
NIM	0.841	-0.023	0.723	0.655	0.865	0.839	-0.063	1					
NOCB	0.712	-0.004	0.751	0.689	0.714	0.694	-0.182	0.735	1				
LCOM	0.044	0.029	0.051	0.021	0.058	0.037	0.069	0.032	0.061	1			
Coh	-0.250	-0.073	-0.309	-0.264	-0.253	-0.227	-0.039	-0.222	-0.266	-0.895	1		
NIC	0.260	-0.013	0.257	0.203	0.441	0.390	-0.115	0.442	0.359	0.027	-0.115	1	
DOIC	0.009	-0.057	0.078	0.073	0.095	0.070	0.073	0.125	0.030	0.233	-0.309	0.529	1

표 3 UIS 그룹에 대한 상관관계

	WMC	NOC	NOM	NOPM	LOC	SLOC	DIT	NIM	NOCB	LCOM	Coh	NIC	DOIC
WMC	1												
NOC	0.147	1											
NOM	0.862	0.201	1										
NOPM	0.757	0.149	0.949	1									
LOC	0.837	0.088	0.659	0.528	1								
SLOC	0.756	0.065	0.569	0.438	0.977	1							
DIT	-0.221	-0.090	-0.319	-0.346	-0.053	0.020	1						
NIM	0.674	0.069	0.542	0.385	0.872	0.867	0.028	1					
NOCB	0.493	0.105	0.340	0.194	0.681	0.670	0.140	0.558	1				
LCOM	0.286	0.058	0.295	0.252	0.243	0.183	0.106	0.149	0.302	1			
Coh	0.431	0.107	0.478	-0.420	-0.336	0.263	0.029	0.218	0.406	0.932	1		
NIC	0.124	-0.043	-0.071	-0.192	0.343	0.310	0.155	0.288	0.440	0.173	-0.120	1	
DOIC	0.013	-0.025	-0.179	-0.269	0.296	0.298	0.286	0.282	0.359	0.061	0.028	0.704	1

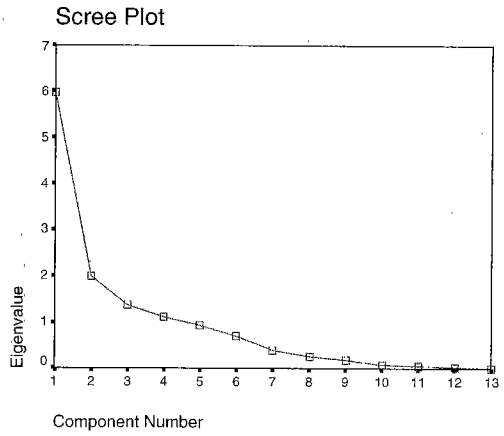

그림 1 AWT 그룹에 대한 Scree 테스트

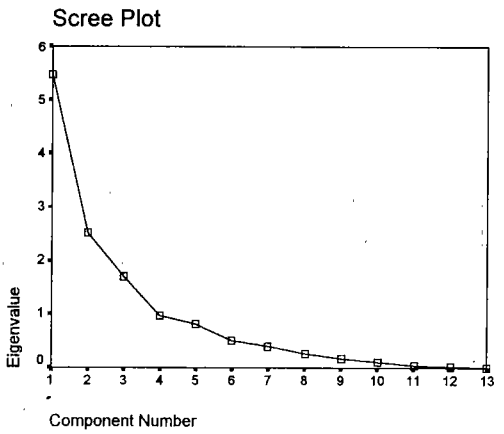

그림 2 UIS 그룹에 대한 Scree 테스트

가 다르기 때문에 차이가 발생하였다.

4.2.3 인자 분석

척도사이에 높은 상관관계를 가지는 것은 같은 근원적인 구조를 측정하는 집합으로 그룹할 수 있다는 것을 가리킨다. 13개의 복잡도 척도를 적은 수의 인자로 분류하기 위해 인자분석을 수행하였다. 인자의 수는 Scree 검정과 고유값(eigenvalue)으로 결정하였다. Scree 검정은 각 인자의 고유값을 Y-축에, 인자의 개수를 X-축으로 표시하여 인자의 수가 증가할수록 고유값이 줄어드는 형태로 보여주는 검정 방식이다[13]. 고유값(eigenvalue)은 각 인자가 전체 분산 중에서 설명하는 부분의 크기를 말하는 것으로 인자가 변수를 몇 개나 설명하는가를 나타내고 보통 고유값이 1이상 되는 것을 인자로 선택한다. 그림 1과 2에서 고유값 1을 기준으로 했을 때 두 그룹 각각 5개의 인자를 추출한다는 것을 보여 준다.

표 4와 5는 두 그룹에 대해 인자 분석한 결과를 보여준다. 인자 분석을 통하여 AWT 그룹에서는 자료 집합 변화의 87.3%를, UIS 그룹에서는 자료 집합 변화의 88.3%를 설명하는 5개의 인자를 확인하였다.

기존의 연구에서와 마찬가지로 첫 번째 인자는 프로그램의 볼륨과 관계된 척도들로 표현되었다. 크기 척도인 WMC, NOM, NOPM, LOC, SLOC와 호출 관계를 나타내는 결합도 척도인 NIM, NOCB가 첫 번째 인자에 포함되고, 두 번째 인자는 응집도 척도인 LCOM과 Coh를 포함하는 것으로 응집도의 결핍을 나타낸다. 세 번째 인자는 내부 클래스 관련 척도인 NIC와 DOIC를 포함하고, 네 번째 인자는 상속 계층의 깊이 척도인 DIT를 포함했다. 마지막으로 다섯 번째 인자는 자식 클래스의 수 척도인 NOC를 포함하였다. 클래스의 크기

척도로 분류되었던 자식 클래스의 수는 다른 크기 척도들과 다른 성질을 가진다는 것을 발견하였다.

두 그룹의 프로그램을 인자 분석한 결과 인자를 나타내는 고유값이 약간 다르기는 하지만 똑같은 인자 5개를 추출하는 것으로 나타났다. 그러므로 Java 프로그램에 대한 복잡도 척도를 설계하기 위해서는 위에서 추출된 5개의 인자를 고려하여야 한다. 응집도 척도인 Coh 값은 두 그룹 모두에서 첫 번째 인자를 구성하는 척도들과 20%정도의 음의 관계를 가지고 있었다. 이것은 응집도가 크기와 관련이 있다는 것을 보여준다.

4.2.4 회귀 분석

기존의 연구에서는 인자의 중요성이나 상관관계에 관계없이 모든 척도들을 대상으로 회귀 분석을 수행하여 인간의 인지 능력을 고려한 소프트웨어의 품질이나 유지보수 노력을 측정하기가 어려웠다. 본 논문에서는 향후 복잡도 척도를 제안할 때 인간의 인지 능력과 인자의 상관관계를 고려한 가중치를 적용하기 위해 인자별로 회귀 분석을 수행하였다. 인자 분석을 수행하면 개개의 인자 값과 추정된 인자 값간의 차이를 제곱한 값이 최소가 되도록 한 인자 점수가 산출된다. 인자 분석에서 인자별로 분류된 척도들은 척도들간의 높은 상관관계로 인해 한 척도의 설명력이 다른 척도에 의해 흡수될 수가 있다. 그럴 경우 추정된 계수가 통계적으로 유의하지 않게 나타날 가능성이 있기 때문에 가장 설명력이 있는 척도들만을 분석에 포함하기 위해 인자 점수를 종속 변수로 하고 각 인자에 포함된 척도들을 독립 변수로 하여 선형 회귀 분석(linear regression analysis)을 수행하였다.

먼저 첫 번째 인자에 대해 모든 해당 척도를 포함시키고 동시에 회귀 계수를 추정하는 동시 투입법(simultaneous method)으로 회귀 분석을 수행한 결과 높은 결정계수(coefficient of determination) R^2(AWT 그룹에서 0.989, UIS 그룹에서 0.961) 값을 가지는 것으로 보아 종속 변수를 테스트에 이용한 척도들로 계산할 수 있다는 것을 보여준다. 표 6에서 보여지는 비표준화 계수(unstandardized coefficients)를 이용하여 첫 번째 인자에 대한 회귀 식을 도출할 수 있고, 표준화 계수로 그룹의 설명력이 가장 높은 척도를 구할 수 있으나 분산확대지수(VIF : Variance Inflation Factor)가 상당히 크다. 분산확대지수는 분석에 사용된 독립 변수 중에서 한 변수를 종속변수로 하고 다른 나머지 변수들을 독립변수로 하여 구한 다중결정계수(R^2)로부터 계산된 $(1-R^2)^{-1}$의 값이다. 따라서 한 변수가 다른 변수와 상관관계가 아주 높을 때 분산확대지수 값이 크게 나온다.

표 4 AWT 그룹의 회전된 인자 행렬

척도 \ 인자	인자 1	인자 2	인자 3	인자 4	인자 5
WMC	0.954	0.044	0.009	0.038	0.007
NOC	0.028	0.028	-0.023	-0.047	0.992
NOM	0.894	0.110	0.005	0.057	0.101
NOPM	0.848	0.084	-0.026	0.075	0.098
LOC	0.933	0.031	0.177	-0.023	-0.005
SLOC	0.923	0.012	0.138	-0.014	-0.015
DIT	-0.022	0.031	-0.004	0.978	-0.047
NIM	0.883	0.012	0.206	-0.064	-0.072
NOCB	0.824	0.093	0.066	-0.233	-0.049
LCOM	-0.014	0.971	0.061	0.021	-0.007
Coh	-0.219	-0.949	-0.114	-0.016	-0.046
NIC	0.306	-0.058	0.847	-0.161	-0.012
DOIC	-0.024	0.238	0.862	0.134	-0.018
Eigenvalue	5.753	1.937	1.576	1.073	1.016
% of variance	44.255	14.898	12.120	8.253	7.813

표 5 UIS 그룹의 회전된 인자 행렬

척도 \ 인자	인자 1	인자 2	인자 3	인자 4	인자 5
WMC	0.869	0.245	-0.077	-0.240	0.065
NOC	0.069	0.041	-0.025	-0.049	0.993
NOM	0.755	0.319	-0.306	-0.366	0.118
NOPM	0.641	0.301	-0.424	-0.409	0.071
LOC	0.950	0.114	0.222	-0.014	0.015
SLOC	0.930	0.036	0.225	0.055	-0.002
DIT	-0.053	0.079	0.099	0.939	-0.041
NIM	0.876	-0.011	0.209	0.047	0.000
NOCB	0.622	0.261	0.401	0.214	0.104
LCOM	0.097	0.961	0.091	0.059	0.005
Coh	-0.236	-0.956	0.008	0.008	-0.050
NIC	0.151	0.119	0.899	-0.021	-0.041
DOIC	0.135	-0.044	0.861	0.172	0.003
Eigenvalue	4.773	2.196	2.152	1.327	1.026
% of variance	36.714	16.894	16.556	10.207	7.890

분산확대지수의 값이 커질 때는 회귀계수의 표준오차가 커지기 때문에 해당 변수를 제거하거나 단계적 변수투입법(stepwise method)을 사용하여 상관관계가 높은 변수들 중 가장 설명력이 있는 독립변수만을 분석에 포함시켜야 한다. 보통 최종 척도 집합의 기준은 분산확대지수 값이 50보다 큰 척도가 존재하지 않는 것이다. 기존 연구에서는 분산확대지수가 50보다 큰 척도를 제거할 때 척도의 수집용이성이나 그 동안의 연구에서 밝혀졌던 몇몇 척도들간의 상관관계에 의해 척도를 소거하였다. 그러나 본 논문에서는 단계적 변수 투입법을 사용

표 6 첫 인자에 대한 개별 척도의 유의성 평가

모델	AWT 그룹			UIS 그룹		
	비표준 화계수	표준화 계수	분산확 대지수	비표준 화계수	표준화 계수	분산확 대지수
상수	-1.056			-1.073		
WMC	0.009215	0.321	17.164	0.008813	0.238	9.669
NOPM	0.01750	0.189	12.350	0.01860	0.248	15.555
NOCB	0.02475	0.142	2.861	0.008970	0.067	2.247
NIM	0.003266	0.138	4.770	0.002864	0.299	5.336
SLOC	0.003003	0.219	57.262	0.008247	0.587	31.331
LOC	-0.0001879	-0.021	55.675	-0.002126	-0.220	52.974
NOM	0.008933	0.114	17.744	-0.003797	0.060	24.295

하여 분산확대지수가 큰 척도를 제거하였다. 표 6에서 LOC 척도는 다른 척도들과 달리 인자 1과 음의 관계를 가지는 것으로 나타났다. 이것은 LOC 척도로는 이 두 그룹에 속한 데이타 집합의 성질을 설명할 수 없다는 것을 의미한다.

단계적 변수투입법을 사용하여 분석한 결과가 표 7에 보여진다. 두 척도가 제외되었음에도 높은 R^2(AWT 그룹에서 0.988, UIS 그룹에서 0.960) 값을 가지는 것으로 나타났다. 이 결과는 인자에 포함된 척도 중 일부만으로도 인자를 설명할 수 있다는 것을 나타낸다. AWT 그룹과 UIS 그룹에 대해 각각의 비표준화계수(unstandardized coefficients)를 이용하여 첫 번째 인자에 대한 회귀식(regression equation)을 도출할 수 있다. 표준화 계수 값으로 미루어 AWT 그룹에서는 WMC 척도가, UIS 그룹에서는 SLOC가 가장 설명력이 있는 척도로 나타났다. 표 6과 비교하여 분산확대지수가 상당히 작아진 것을 알 수 있다. WMC와 SLOC의 상관관계가 상대적으로 다른 척도에 비해 높다는 것을 나타내고 있다.

표 8에 회귀 분석에서 모형의 적합도를 검정해주는 각 그룹의 결정계수 R^2가 인자들에 대해 나타난다. 이것은 해당 인자들을 구성하고 있는 척도들로 나타내는 것이 신뢰성이 있다는 것을 나타낸다.

표 7 단계적 변수투입법을 사용하여 분석한 결과

모델	AWT 그룹			UIS 그룹		
	비표준화 계수	표준화 계수	분산확 대지수	비표준화 계수	표준화 계수	분산확 대지수
상수	-1.056			-1.074		
WMC	1.027E-2	0.357	15.335	6.522E-3	0.176	5.007
NOPM	2.533E-2	0.273	2.596	1.499E-2	0.200	2.702
NOCB	2.695E-2	0.155	2.639	6.597E-3	0.049	1.892
NIM	3.384E-3	0.143	4.098	2.572E-3	0.268	4.060
SLOC	2.409E-3	0.176	13.257	6.243E-3	0.444	6.297

표 8 인자들의 결정계수

결정계수 / 인자	R^2	
	AWT 그룹	UIS 그룹
인자 1	0.988	0.960
인자 2	0.975	0.951
인자 3	0.955	0.912
인자 4	0.957	0.881
인자 5	0.983	0.986

표 9 나머지 인자에 대한 개별 척도의 유의성 평가

모델		AWT 그룹			UIS 그룹		
		비표준 화계수	표준화 계수	분산확 대지수	비표준 화계수	표준화 계수	분산확 대지수
인자 2	상수	-1.145			-0.541		
	LCOM	1.720	0.608	5.015	1.510	0.535	7.567
	Coh	-1.471	-0.405	5.015	-1.499	-0.457	7.567
인자 3	상수	-0.467			-0.505		
	NIC	1.035	0.575	1.389	0.257	0.582	1.983
	DOIC	1.379	0.542	1.389	0.891	0.451	1.983
인자 4	상수	-0.924		1	-1.535		1
	DIT	1.955	0.978		2.106	0.939	
인자 5	상수	-0.150		1	-0.181		1
	NOC	0.229	0.992		0.151	0.993	

표 9에 나머지 인자들에 대한 분석 결과가 나타난다. 네 개의 인자들에 대해서는 다중공선성이 나타나지 않았다.

표 7과 9에 나타난 비표준화계수(unstandardized coefficients)를 이용하여 각 인자에 대한 회귀식을 도출할 수 있다.

5. 평 가

추출한 회귀식을 평가하기 위해 교차 검증(cross-validation)을 실시하였다. AWT 그룹의 데이타에 대한 인자 점수를 예측하기 위해 UIS 그룹의 회귀식을 사용하고, UIS 그룹의 데이타에 대한 인자 점수를 예측하기 위해 AWT 그룹의 회귀식을 사용하여 서로 교차 타당성 검증을 수행하였다. 먼저 다음의 첫 번째 인자 회귀식 (3)과 (4)에 대해 수행한 결과 AWT 그룹에서 0.981, UIS 그룹에서 0.966의 높은 상관관계를 가지는 것을 그림 3과 4에서 볼 수 있다. 이것은 회귀식이 타당하다는 것을 보여주고 있다. 각 그룹의 첫 번째 인자 점수에 대한 근사치를 나타내기 위해 $\widehat{F1}_{AWT}$과 $\widehat{F1}_{UIS}$을 사용하였다.

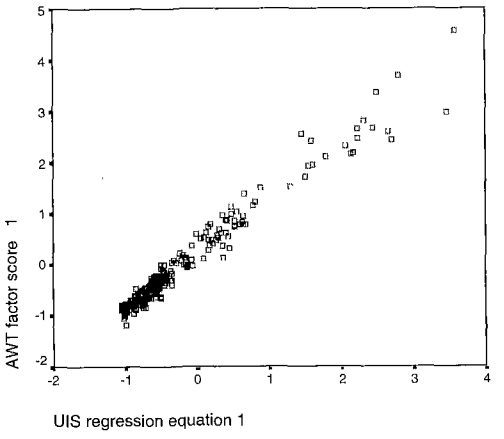

그림 1 AWT 그룹의 인자 1과 UIS 그룹의 회귀식 1의 상관관계

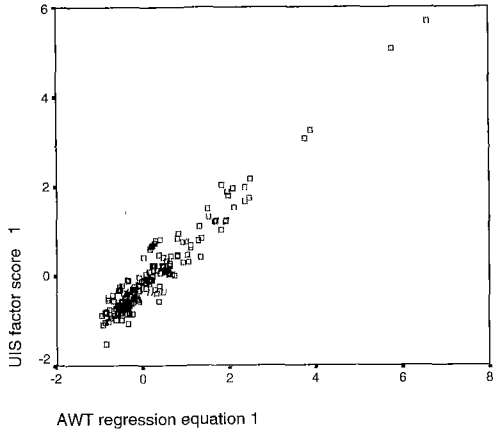

그림 2 UIS 그룹의 인자 1과 AWT 그룹의 회귀식 1의 상관관계

$$\widehat{F1}_{AWT} = -1.056 + 1.027E-2WMC + 2.533E-2NOPM + 2.695E-2NOCB + 3.384E-3NIM + 2.409E-3SLOC \quad (3)$$

$$\widehat{F1}_{UIS} = -1.074 + 6.522E-3WMC + 1.499E-2NOPM + 6.597E-3NOCB + 2.572E-3NIM + 6.243E-3SLOC \quad (4)$$

$$\widehat{F2}_{AWT} = -1.145 + 1.720LCOM - 1.471Coh \quad (5)$$

$$\widehat{F2}_{UIS} = -0.541 + 1.510LCOM - 1.499Coh \quad (6)$$

두 번째 인자 회귀식 (5)과 (6)에 대해 AWT 그룹에서 0.987, UIS 그룹에서 0.975로 높은 상관관계를 나타내는 것을 그림 5와 6에서 보여준다.

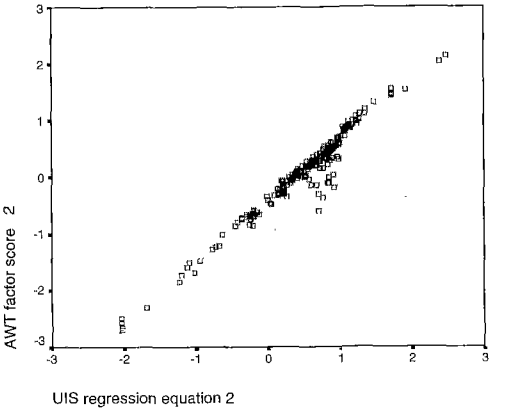

그림 5 AWT 그룹의 인자 2와 UIS 그룹의 회귀식 2의 상관관계

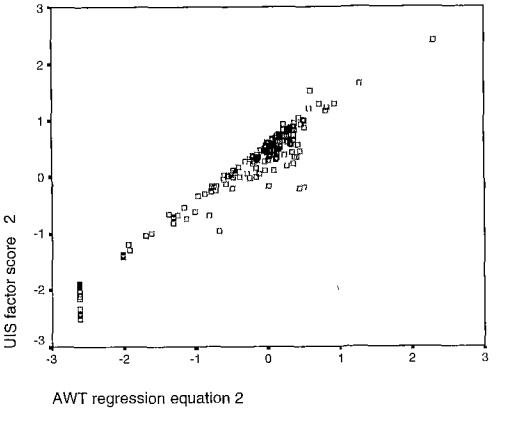

그림 6 UIS 그룹의 인자 2와 AWT 그룹의 회귀식 2의 상관관계

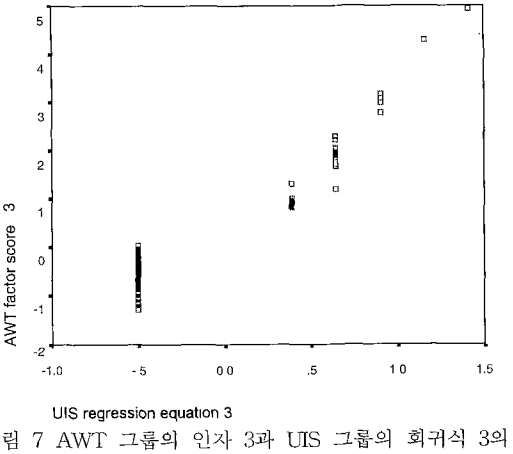

그림 7 AWT 그룹의 인자 3과 UIS 그룹의 회귀식 3의 상관관계

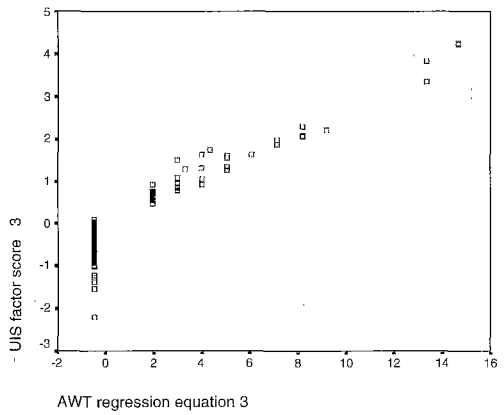

그림 8 UIS 그룹의 인자 3과 AWT 그룹의 회귀식 3의 상관관계

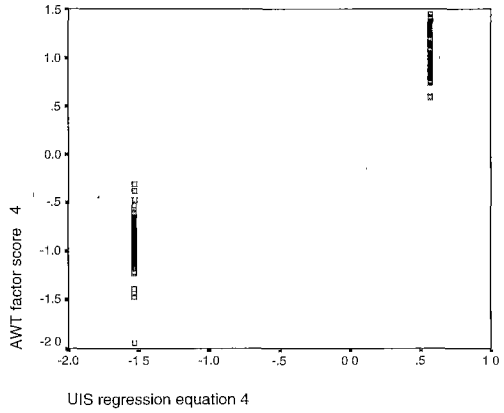

그림 9 AWT 그룹의 인자 4와 UIS 그룹의 회귀식 4의 상관관계

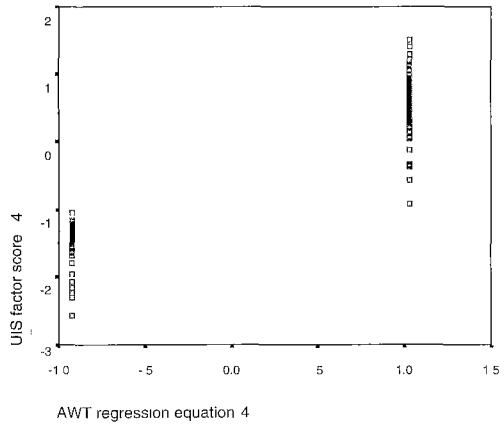

그림 10 UIS 그룹의 인자 4와 AWT 그룹의 회귀식 4의 상관관계

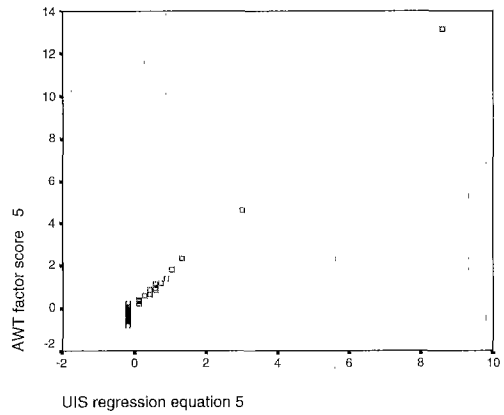

그림 11 AWT 그룹의 인자 5와 UIS 그룹의 회귀식 5의 상관관계

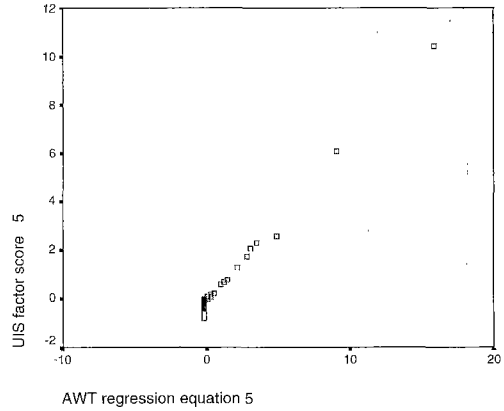

그림 12 UIS 그룹의 인자 5와 AWT 그룹의 회귀식 5의 상관관계

$$\widehat{F3}_{AWT} = -0.467 + 1.035NIC + 1.379DOIC \quad (7)$$

$$\widehat{F3}_{UIS} = -0.505 + 0.257NIC + 0.891DOIC \quad (8)$$

세 번째 인자 회귀식 (7)과 (8)에 대해 AWT 그룹에서 0.949, UIS 그룹에서 0.941의 높은 상관관계를 나타냈다. 내부 클래스를 사용하지 않는 데이타로 인해 그림 7과 8에서 보는 바와 같이 0이하의 값이 많이 나타났다.

$$\widehat{F4}_{AWT} = -0.924 + 1.955DIT \quad (9)$$

$$\widehat{F4}_{UIS} = -1.535 + 2.106DIT \quad (10)$$

네 번째 인자 회귀식 (9)와 (10)에 대해 AWT 그룹에서 0.978, UIS 그룹에서 0.939에 대해서도 높은 상관관계를 가지지만 상속 계층의 깊이가 단지 두 개만 존재하는 관계로 그림 9와 10의 결과를 보여주고 있다.

$$\widehat{F5}_{AWT} = -0.150 + 0.229NOC \quad (11)$$

$$\widehat{FS}_{UIS} = -0.181 + 0.151NOC \quad (12)$$

다섯 번째 인자 회귀식 (11)과 (12)에 대해 AWT 그룹에서 0.992, UIS 그룹에서 0.993의 높은 상관관계를 나타냈다. 자식 클래스의 사용이 거의 없다는 것을 그림 11과 12를 통해 알 수 있다.

이러한 위의 결과들은 각각의 회귀식들이 인자를 정확하게 반영하고, 모집단 전체에 대해 높은 신뢰성을 가진다는 것을 가리킨다.

6. 결 론

본 논문에서는 몇몇 선행 연구에서 제시한 객체지향 설계 척도들과 Java 언어의 특징인 내부 클래스(inner class) 관련 척도, 그리고 크기 척도의 관련성을 파악하기 위해 포괄적인 실험적 검증을 수행하였다. 이 논문에서 연구된 13개의 척도들 중 일부가 Java 프로그램에서 서로 관계가 있다는 것을 밝혔다. 이것은 많은 척도들이 본질적으로 자료의 유사한 면을 나타내는 것으로 측정된다. 사실, 실제로 척도들에 의해 나타난 인자 수는 척도들의 수보다 훨씬 낮았다. 이것은 문헌에서 제안된 많은 척도들이 유사한 생각과 가정에 기초한다는 사실을 나타낸다.

기존의 연구에서 클래스의 크기만이 주요 인자로 고려되었던 것과는 달리 클래스의 크기 외에도 메소드 호출 빈도, 응집도, 자식 클래스의 수, 내부 클래스와 상속 계층의 깊이가 서로 다른 성질을 가지는 중요한 복잡도 인자라는 것을 제안하였다. 또한 같은 인자에 속한 척도들이 매우 높은 상관관계를 가지므로 인자에 포함된 척도 중 일부만으로도 인자를 설명할 수 있다는 것을 발견하였다. 이것은 많은 척도를 추가 분석하는 경우에도 같은 성질을 가지는 척도는 생략이 가능하다는 것을 의미한다. 한편 응집도가 크기 및 결합도와 음의 관계를 가진다는 것을 발견하였다. 이것은 프로그램의 크기가 커지고 결합도가 높아질수록 응집도가 낮아진다는 것을 나타낸다. 이러한 점들은 복잡도 척도를 제안할 때 인간의 인지 능력이나 인자의 중요성과 상관관계가 반영되어야 한다는 것을 나타낸다. 그러므로 기존의 연구에서 모든 분석 척도를 대상으로 한 회귀 분석은 소프트웨어 품질이나 유지 보수 노력을 정확하게 반영하지 못한다고 할 수 있다.

본 논문에서는 인자 분석을 통하여 얻은 인자 점수를 가지고 회귀 분석을 수행하여 각 인자에 대한 회귀식을 도출하였고, 각각의 회귀식을 두 그룹에 교차 적용하여 회귀식의 타당성을 검증하였다. 향후 연구과제로는 검증된 인자들을 이용하여 Java 언어의 품질을 측정할 수 있는 복잡도 척도를 제안하는 것이 필요하고, 보완 연구로 분석에 사용된 대부분의 프로그램이 100 SLOC 이하였던 관계로 규모가 큰 프로그램을 수집하여 프로그램 크기별로 유형을 비교하는 연구 등이 필요하다.

참 고 문 헌

[1] Mohammad A., "Putting Metrics into Software Perspectives," http://www.cs.usask.ca/homepages/grads/moa135/856/metrics/metrics.html, Oct. 1995.

[2] Chidamber S. R. and Kemerer C. F., "A Metrics Suite for Object-Oriented Design," *IEEE Trans. on Software Engineering*, Vol. 20, No. 6, pp. 476-493, Jun. 1994.

[3] Fenton N., *Software Metrics - A Rigorous Approach*, Chapman and Hall, 1991.

[4] Hitz M., Montazeri B., "Measuring Coupling and Cohesion in Object-Oriented Systems," in *Proc. Int. Symposium on Applied Corporate Computing*, Monterey, Mexico, Oct. 1995.

[5] Bieman J. M., Kang B. K., "Cohesion and Reuse in an Object-Oriented System," in *Proc. ACM Symp. Software Reusability(SSR'94)*, pp. 259-262, 1995.

[6] Henderson-sellers B., "Object-Oriented Metrics: Measures of Complexity," Prentice-Hall, Hemel Hempstead, UK, 1996.

[7] Briand L., Morasca S., and Basili V., "Property Based Software Engineering Measurement," *IEEE Trans. on Software Engineering*, Vol. 22, No. 1, pp. 68-86, Jan. 1996.

[8] Ramon A. Mata-Toledo, David A. Gustafson, "A Factor Analysis of Software Complexity Measures," *Journal of Systems and Software*, Vol. 17, pp. 267-273, 1992.

[9] Daniel Coupal and Pierre N. Robillard, "Factor Analysis of Source Code Metrics," *Journal of Systems and Software*, Vol. 12, pp. 263-269, 1990.

[10] Li Wei, Henry Sallie, "Object-Oriented Metrics that Predict Maintainability," *Journal of Systems and Software*, Vol. 23, pp. 111-122, 1993.

[11] Samuel B. Green, Neil Salkind, Teresa Akey, and Neil J. Salkind, *Using SPSS for Windows: Analyzing and Understanding Data*, Upper Saddle River, NJ: Prentice Hall, 1997.

[12] Lakshmanian K. B., Jayaprakash S., and Sinha P. K., "Properties of Control-Flow Complexity Measures," *IEEE Trans. on Software Engineering*, Vol. 17, No. 12, pp. 1289-1295, Dec. 1991.

[13] Tian J. and Zelkowitz M. V., "A Formal Program Complexity Model and Its Application," *Journal of*

Systems and Software, Vol. 17, pp. 253-266, 1992.
[14] Weyuker E. J., "Evaluating Software Complexity Measures," *IEEE Trans. on Software Engineering*, Vol. 14, No. 9, pp. 1357-1365, Sept. 1988.
[15] Roger S. Pressman, *Software Engineering - A Practitioner's Approach*, McGraw-Hill, 1998.
[16] Briand L., Daly J., and Wust J., "A Unified Framework for Cohesion Measurement in Object-Oriented Systems," *Empirical Software Engineering Journal*, 3(1), pp. 65-117, 1998.
[17] McCabe, T. J., "A Complexity Measure," *IEEE Trans. on Software Engineering*, Vol. 2, No. 4, pp. 308-320, April 1976.
[18] Brian W. Bush, *BANDA Java Packages*, http://www.sladen.com/Java.
[19] Andrew Cain, Rajesh Vasa, *Java Metric Analyser*, http://www.csse.swin.edu.au/cotar/jmetric/index.html.

김 재 웅
1993년 2월 전북대학교 전자계산학과 졸업(이학사). 1995년 2월 전북대학교 전산통계학과 졸업(이학석사). 1995년 3월 ~ 현재 전북대학교 전산통계학과 박사과정. 1995년 3월 ~ 1996년 3월 군산대학교 계산통계학과 조교. 관심분야는 소프트웨어공학, 소프트웨어 척도, 컴포넌트 복잡도, 인지과학

유 철 중
정보과학회논문지 : 소프트웨어 및 응용
제 27 권 제 7 호 참조

장 옥 배
정보과학회논문지 : 소프트웨어 및 응용
제 27 권 제 7 호 참조

<부록>

AWT GROUP 측정값(일부)

척도 / 프로그램	LOC	SLOC	LCOM	NOM	NOPM	NIC	DOIC	NOCB	NOC	DIT	NIM	WMC	COH
1	601	388	0.99	13	11	0	0	7	0	1	164	228	0.086
2	462	365	0.94	23	18	0	0	20	0	0	61	143	0.100
3	567	349	1.00	37	15	2	1	18	5	1	125	137	0.027
4	504	312	0.94	74	47	0	0	32	0	1	196	196	0.073
5	506	290	0.90	23	13	0	0	16	0	0	247	117	0.141
6	415	286	0.97	27	26	0	0	15	0	1	65	122	0.065
7	473	256	0.93	34	28	4	1	35	0	0	152	102	0.098
8	462	252	0.97	46	37	2	1	13	0	0	164	102	0.051
9	349	251	0.78	38	32	0	0	15	0	1	83	118	0.243
10	416	248	0.98	70	53	1	1	34	0	0	184	127	0.034
11	374	239	0.95	29	25	2	1	9	0	1	248	99	0.083
12	338	235	0.86	51	46	0	0	17	2	0	131	122	0.157
13	342	230	0.99	37	33	0	0	18	0	0	75	95	0.036
14	400	229	0.96	52	50	0	0	28	0	1	149	125	0.058
15	313	195	0.62	28	24	0	0	9	3	1	54	92	0.405
16	280	192	0.94	42	37	0	0	15	4	0	61	93	0.083
17	341	185	0.97	40	34	2	1	12	0	0	116	80	0.055
18	329	182	0.91	42	40	0	0	11	10	0	36	118	0.113
19	282	172	0.97	9	6	3	1	13	0	0	32	16	0.138
20	224	166	0.80	21	20	0	0	11	0	0	27	41	0.238
							⋮						
							⋮						

UIS GROUP 측정값(일부)

척도 / 프로그램	LOC	SLOC	LCOM	NOM	NOPM	NIC	DOIC	NOCB	NOC	DIT	NIM	WMC	COH
1	626	383	0.96	100	69	0	0	26	22	0	685	190	0.049
2	557	368	0.98	85	60	0	0	17	0	0	727	145	0.031
3	425	276	0.94	78	73	0	0	13	0	0	165	148	0.072
4	351	260	1.23	2	2	7	1	17	0	1	112	34	0.385
5	341	259	0.96	20	10	4	1	33	0	1	282	30	0.088
6	314	257	0.95	26	24	0	0	19	0	1	309	34	0.085
7	363	250	0.94	32	9	6	1	40	0	1	314	54	0.089
8	384	234	0.94	63	49	0	0	14	0	0	431	113	0.074
9	316	231	0.92	28	6	2	1	29	0	1	311	45	0.113
10	379	226	0.65	7	5	0	0	14	0	1	321	72	0.446
11	289	219	0.93	4	4	3	1	20	0	1	290	40	0.303
12	308	218	0.86	15	12	0	0	38	0	1	14	62	0.197
13	278	209	0.91	29	25	0	0	34	0	1	260	46	0.122
14	229	185	0.76	20	14	0	0	10	0	0	48	72	0.278
15	251	177	0.90	10	6	4	1	33	0	1	212	52	0.189
16	311	169	1.00	22	2	12	2	33	0	1	244	79	0.045
17	311	169	1.00	22	2	12	2	33	2	1	244	77	0.045
18	258	166	0.97	42	28	0	0	14	0	0	324	55	0.053
19	196	166	0.00	1	1	1	1	8	0	1	206	2	1.000
20	223	161	0.93	21	16	0	0	21	0	1	93	67	0.115
							⋮						
							⋮						