Java 프로그램에 적용한 객체지향 척도

김재웅* , 유철중** , 장옥배**
* 한려대학교 정보통신학과
** 전북대학교 컴퓨터과학과

Object-Oriented Measures for Java Program

Jae-Woong Kim* , Cheol-Jung Yoo** , Ok-Bae Chang**
* Dept. of Information & Telecommunication, Hanlyo University
** Dept of Computer Science, Chonbuk National University

요 약

다양한 소프트웨어 척도들이 절차적 패러다임에서 유용하다는 것이 밝혀졌고 객체지향 패러다임에 대해서도 많은 설계 척도들이 언어 독립적으로 제안되었다. 언어 독립적인 척도로부터 특정 프로그래밍 언어에 대한 척도를 명확하게 하는 것이 중요한데도 불구하고 Java 언어에 대한 척도는 거의 제안되지 않고 있는 형편이다. 따라서 본 논문에서는 Briand가 제안한 속성을 만족하는 척도들과 Java 언어의 특징인 내부 클래스를 반영한 척도와 크기 척도 등 13개의 척도를 Java 프로그램에 적용하여 척도들 사이의 관계를 분석하였다. 클래스의 크기와 메소드 호출 빈도, 응집도, 자식 클래스의 수, 내부 클래스와 상속 계층의 깊이가 주요 인자라는 것을 보여준다. 또한 응집도가 다른 척도들과 음의 관계를 가진다는 것이 발견되었다. 보다 적은 척도를 가지고 인자를 설명할 수 있는 회귀식을 도출하고 교차검증을 실시하였다.

1. 서론

오늘날 소프트웨어 공학자에게 가장 중요한 관심사는 소프트웨어 시스템을 유지보수하는데 사용되는 비용에 있다. 소프트웨어 유지보수 활동에 대한 비용이 개발비용을 초과하는 것을 쉽게 관찰할 수 있다. 유지보수의 높은 비용은 생명주기의 다른 단계보다 유지보수 단계의 제어에 더 큰 어려움을 준다. 유지보수 비용을 줄이기 위해서는 개발 주기의 초기 단계에서부터 시스템의 품질을 평가할 필요가 있다. 완전한 소스코드나 설계 표현으로부터 계산된 소프트웨어 복잡도 척도가 정량적인 정보를 제공함으로써 유지보수 비용을 감소할 수 있다.

다양한 객체지향 척도가 상속 트리나 메소드 호출, 응집도와 결합도를 비롯한 다양한 설계 매개변수를 사용하여 객체지향 설계의 품질을 평가하기 위해 제안되었다. 그러나 어떤 척도가 실제로 중요한 품질의 면을 나타내는 지를 우리는 알 수가 없다. 많은 주위의 척도들은 다른 척도나 속성을 명확하게 분류하는데 실패한 것에서 기인한다. 많은 연구에서 소프트웨어 척도들 사이에 높은 상관관계가 있다는 것을 말한다. 그러므로 우선 척도들간의 관련성을 규명하는 것이 가장 의미있는 척도를 선택하는데 있어서 중요하다.

설계 척도들은 구현이 시작하기 전에 설계에서 문제를 탐지하기 위해 개발과정의 초기단계에 적용할 수 있다. 그러나 구현이 시작되기 전에 측정이 발생한다면 메소드 호출 등에 관련된 초기 설계 정보가 불확실하기 때문에 다른 측정자료를 얻게 될 수 있다.

본 논문에서는 Briand[1]가 제안한 속성을 만족하는 척도들과 내부 클래스를 반영한 척도와 크기 척도 등 13개의 척도를 인터넷과 JDK 12에서 수집한 Java 소스 프로그램에 적용하여 척도들 사이의 관계를 분석한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 측정할 척도를 정의하고, 3장에서는 자료 분석 환경과 분석 결과를 설명하고, 마지막으로 4장에서 결론을 기술한다.

2. 소프트웨어 척도

이 논문에서는 Briand[1]가 제시한 척도들의 속성을 만족하는 객체지향 설계 척도들과 내부 클래스를 고려한 척도 그리고 크기 척도 등 13개의 척도를 평가한다.

2.1 측정 소프트웨어 척도

WMC(Weighted Methods per Class) 척도[7]는 클래스에 정의된 메소드의 복잡도를 측정하는 것이다. WMC는 메소드에 대한 McCabe[8]의 사이클로메틱 복잡도 값의 합으로 계산하였다. NOC(Number Of Children) 척도는 한 클래스가 가지고 있는 직접적인 자식 클래스의 수이다. NOM(Number Of Methods)와 NOPM((Number of .Public Methods) 척도는 각각 메소드의 수와 공용 메소드의 수이다. LOC(Line Of Code) 척도는 프로그램 헤더, 선언문, 실행문과 비실행문 모두를 포함하고 주석을 제외한 라인 수를 측정한다. SLOC(Statements) 척도는 실행문만을 고려하는 것으로 클래스 안의 세미콜론의 수를 측정한다.

DIT(Depth of Inheritance Tree) 척도는 상속 트리의 노드로부터 루트까지의 최대 길이이다 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. NOCB(Number Of CollaBorator) 척도는 메소드에서 참조하는 collaborator의 수를 측정한다

LCOM(Lack of COhesion in Methods) 척도는 클래스의 응집도 결핍을 측정하는 것이다. LCOM은 Henderson -Sellers[9]가 정의한 식 (1)을 사용하였다

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

Coh[10] 척도는 LCOM의 변형으로 클래스의 응집도를 측정하는 척도로 Briand[1]의 속성을 모두 만족하였다.

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

두 척도 모두 클래스의 메소드에서 사용한 전체 변수 사용수를 중심으로 응집도를 측정하였다

NIC(Number of Inner Class) 척도는 내부 클래스의 수를 측정한다 DOIC(Depth Of Inner Class) 척도는 내부 클래스의 상속 깊이를 계산한다.

3. 데이터 분석 및 평가

2장에서 살펴본 척도들을 두 그룹의 프로그램에 각각 적용하였다. 첫 번째 그룹은 정형화되었다고 할 수 있는 SUN JDK 1.2를 구성하는 awt 패키지에서 220개의 프로그램(AWT)을 사용하였고, 두 번째 그룹은 인터넷에서 수집한 사용자 인터페이스와 관련되어 구현된 177개의 Java 프로그램(UIS)을 사용하여 척도 값을 추출하였다. 두 그룹 모두 500 SLOC 이하의 프로그램을 대상으로 하였다 척도 값의 측정은 Banda의 Java Source Metrics[11]와 Andrew와 Rajesh의 JMetric[12] 등의 자동화 도구를 주로 사용하였다.

3.1 데이터 분석 방법

397개의 Java 프로그램에 대해 수집된 크기, 결합도, 응집도와 상속 척도 자료를 분석하기 위해 기술 통계 분석, 상관관계 분석, 인자 분석, 회귀 분석을 사용하였다.

인자 분석(Factor Analysis)은 여러 변인들간의 상호 관계로부터 공통 요소를 구하고 측정치의 중복성을 찾아내어 몇 개의 기본적인 인자를 추출하는 기법이다. 인자 분석에서 얻은 인자 점수를 다중 회귀 분석에서 사용하였다.

다중 회귀 분석(Multiple Regression Analysis)은 하나이상의 독립변수와 종속변수와의 관계를 선형 관계식으로 나타낸다. 변수들과의 상호관계를 분석하고 독립변수의 변화로부터 종속변수의 변화를 예측하기 위해 사용하였다.

3.2 분석 결과

이 절에서는 통계 분석 결과를 보여주고, 추출된 회귀식을 이용하여 두 그룹에 대해 교차 검증한 결과를 나타낸다.

3.2.1 기술 통계

표 1은 AWT와 UIS에 대한 기술 통계 결과를 제공한다.

표 1 두 그룹에 대한 기술 통계

척도	AWT		UIS	
	평균	표준편차	평균	표준편차
WMC	29.61	34.80	25.81	27.02
NOC	0.65	4.32	1 20	6.56
NOM	13.85	12.78	13.43	15.70
NOPM	11.18	10.79	10.69	13.34
LOC	97.85	113.50	115.53	103.41
SLOC	58.42	72.96	73.56	71.13
DIT	0.47	0.50	0.73	0.45
NIM	30.45	42.18	83.31	104.35
NOCB	8.36	5.74	10.95	7.43
LCOM	0.88	0.35	0.729	0.354
Coh	0.248	0 275	0.374	0.305
NIC	0 15	0.52	0.93	2.26
DOIC	0.23	0.42	0.30	0.51

AWT 패키지의 프로그램을 분석한 결과 표 1에서 보는 것처럼 메소드 호출과 관련 있는 NIM 측정값이 다른 언어에 비해 상당히 많은 것을 알 수 있다. 객체지향 원칙에 충실하였을 경우에 메소드 호출이 많아진다는 것과 관련이 있다[7]. 또한 상속이 거의 없고, 클래스가 적은 수의 자식 클래스(NOC)를 가진다는 것을 가리킨다. 응집도와 관련해서 척도 LCOM과 Coh를 분석한 결과 대부분의 프로그램들이 낮은 응집도를 가진다는 것을 보여준다 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 보여준다. 사용자 인터페이스와 관련 있는 프로그램이어서 결합도 척도 값이 높게 나타나는 것으로 생각되고, 또한 내부 클래스의 사용도 첫 번째 그룹보다 많았다.

3.2.2 인자 분석

표 2와 3은 두 그룹에 대해 회전한 인자 행렬이다.

표 2 AWT의 회전된 인자 행렬

	FACTOR 1	FACTOR 2	FACTOR 3	FACTOR 4	FACTOR 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 3 UIS의 회전된 인자 행렬

	FACTOR 1	FACTOR 2	FACTOR 3	FACTOR 4	FACTOR 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

인자 분석을 통하여 AWT에서는 자료 집합 변화의 87.3%를, UIS에서는 자료 집합 변화의 88.3%를 설명하는 5개의 인자를 확인하였다. 첫 번째 인자는 크기 척도인 WMC, NOM, NOPM, LOC, SLOC와 호출 관계를 나타내는 결합도 척도인 NIM, NOCB를 포함하고, 두 번째 인자는 응집도 척도인 LCOM과 Coh를 포함하는 것으로 응집도의 결핍을 나타낸다. 세 번째 인자는 내부 클래스 관련 척도이고, 네 번째 인자는 상속 계층의 깊이와 관계있다. 마지막으로 다섯 번째 인자는

자식 클래스의 수를 나타내는 인자이다.

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 가리킨다.

3.2.3 회귀 분석

인자 분석을 통하여 추출된 5개의 인자에 대한 식을 얻기 위해 회귀 분석을 사용하였다. 인자 분석에서 얻은 각각의 인자 점수를 종속 변수로 하고 인자에 해당되는 척도를 독립 변수로 하여 선형 회귀 분석을 수행하였다. 회귀 분석을 수행한 결과 식(3)에서 (12)까지의 회귀식을 얻었다.

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

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

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

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

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

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

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

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

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

$$\widehat{F5}_{UIS} = -0.181 + 0.151NOC \quad (12)$$

3.2.4 평가

: 추출한 회귀식을 평가하기 위해 교차 검증을 실시하였다. AWT의 데이터에 대한 인자 점수를 예측하기 위해 UIS의 회귀 식을 사용하고, UIS의 데이터에 대한 인자 점수를 예측하기 위해 AWT의 회귀 식을 사용하여 서로 교차 타당성 검증을 수행하였다. 표 4에서 두 그룹의 회귀식을 교차 검증한 상관관계 결과를 보여준다. 모든 인자에 대해 높은 상관관계를 가지는 것으로 나타났다.

표 4 두 그룹의 회귀식 교차 검증 상관관계 분석

	상관계수	
	AWT	UIS
Factor 1	0.981	0.966
Factor 2	0.987	0.975
Factor 3	0.949	0.941
Factor 4	0.978	0.939
Factor 5	0.992	0.993

이러한 위의 결과들은 모든 회귀식들이 정확하고, 모집단 전체에 대해 높은 신뢰성을 가진다는 것을 가리킨다.

4. 결론

우리는 몇몇 참고문헌에서 제시한 객체지향 설계 척도들과 Java 특성 척도, 크기 척도의 포괄적인 경험적 확인을 수행하였다. 이 논문에서 연구된 크기, 결합도, 응집도와 상속에 관련된 13개의 척도들이 대부분 자료의 유사한 면을 나타내는 것으로 밝혀졌다. 사실, 실제로 척도들에 의해 나타난 인자 수는 척도들의 수보다 훨씬 낮았다. 이것은 문헌에서 제안된 많은 척도들이 유사한 생각과 가정에 기초한다는 사실을 나타낸다

두 그룹의 프로그램을 분석하여 추출된 척도 값을 통계 분석을 통해 클래스의 크기와 메소드 호출 빈도, 응집도, 자식 클래스의 수, 내부 클래스와 상속 계층의 깊이가 중요한 품질 인자라는 것을 보여준다. 한편 응집도가 다른 척도들과 음의 관계를 가진다는 것을 발견하였다. 회귀 분석을 통하여 얻은 회귀식을 통하여 각 인자에 대한 예측 방정식을 도출하였고, 각각의 회귀식을 두 그룹에 교차 적용하여 회귀식의 타당성을 검증하였다.

이 논문에서 수행된 연구 결과를 바탕으로 예측 방정식을 표준화하고, 그 예측 방정식을 이용하여 Java 언어의 품질을 측정할 수 있는 복잡도 척도를 제안하는 것이 필요하다. 분석에 사용된 대부분의 프로그램이 100 SLOC 이하였던 관계로 큰 프로그램을 더 수집하여 프로그램 크기 별로 유형을 비교하는 연구가 필요하다.

참고문헌

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

[2] Lakshmanian K. B., Jayaprakash S,, and Sinha P. K., "Properties of control-flow complexity measures," *IEEE Trans Software Engineering*, vol. 17, no. 12, pp. 1,289-1,295, Dec. 1991.

[3] 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.

[4] Weyuker E. J., "Evaluating software complexity measures," *IEEE Trans. Software Engineering*, vol. 14, no. 9, pp. 1,357-1,365, Sept. 1988.

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

[6] Roger S. Pressman, *Software Engineering - A Practitioner's Approach*, McGraw-Hill, 1998.

[7] Chidamber S R and Kemerer C. F. , "A Metrics Suite for Object-Oriented Design," *IEEE Trans. Software Engineering*, vol. 20, no. 6, pp. 476-493, June 1994.

[8] McCabe, T. J., "A Complexity Measure," *IEEE Trans. on Software Engineering*, vol. 2, no 4, pp. 308-320, April 1976.

[9] Henderson-sellers B., "Object-Oriented Metrics: measures of complexity," Prentice Hall, Hemel Hempstead, UK, 1996.

[10] 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.

[11] Brian W. Bush, *BANDA Java Packages*, http://www.sladen.com/Java

[12] Andrew Cain, Rajesh Vasa, *Java Metric Analyser*, http://www csse swin edu au/cotar/jmetric/ index.html.

[13] 유대근, 권영식, 통계분석을 위한 SPSSWIN 8.0, 기한재, 1999

[14] Briand L., Emam K. El, and Morasca S., "Theoretical and Empirical Validation of Software Product Measures," ISERN Technical Report 95-03, 1995.

[15] Korson T, Mcgregor J. D., "Understanding Object-Oriented: A Unifying Paradigm," *Communicat- ion ACM*, 33, pp. 41-60, 1990