프로그램 볼륨이 복잡도에 미치는 영향 분석

김재웅[U] 유철중 장옥배
전 북 대 학 교 컴 퓨 터 과 학 과
jwkim@cs.chonbuk.ac.kr, {cjyoo, objang}@moak.chonbuk.ac.kr

Effect Analysis of Program Volume on Complexity

Jae-Woong Kim[U] Cheol-Jung Yoo Ok-Bae Chang
Dept. of Computer Science, Chonbuk National University

요 약

최근 객체지향 소프트웨어 개발이 일반화되면서 품질 좋은 객체지향 소프트웨어의 개발을 돕기 위한 목적으로 객체지향 소프트웨어의 복잡도 척도에 관한 연구들이 다양하게 행해져 왔다. 대부분의 연구가 구조적 복잡도 측정에 중점을 두어 프로그램 크기와 관련된 요인들을 측정하는 척도들이 제시되었다. 한편 소프트웨어 개발이나 복잡도 측정에 대하여 인지 심리 이론을 적용하여, 인간의 단기 기억의 한계 7±2와 제어 논리 처리에 3±1을 고려한 연구들이 많이 행해졌다. 본 논문에서는 프로그램 볼륨과 복잡도의 관계를 조사하기 위해 13개 척도의 값을 추출한 후 통계적 분석을 수행하고, 인지 심리 이론과의 관계를 파악해 보았다.

1. 서론

소프트웨어의 규모와 복잡성이 증대되면서 소프트웨어 위기를 맞게 되었는데, 이를 해결하기 위해 소프트웨어 개발 방법이 절차지향 방법에서 객체지향 기법으로 전환되었다. 객체지향 기법의 사용으로 인해 코드 재사용율과 유지보수 및 시스템 확장이 용이해져 소프트웨어 생산성 증가를 가져왔다. 최근 객체지향 소프트웨어 개발이 일반화되면서 품질 좋은 객체지향 소프트웨어의 개발을 돕기 위한 목적으로 다양한 연구들이 행해져 왔다. 그 중 하나가 객체지향 소프트웨어의 복잡도 척도에 관한 연구들이다. 소프트웨어 복잡도에는 여러 가지 유형이 존재하는데, 대부분의 연구가 구조적 복잡도 척도에 중점을 두어왔다. 구조적 복잡도를 측정하는 척도에는 모듈 척도(예, 크기 척도, 응집도 척도, 제어 흐름 복잡도 척도)와 모듈간(intermodule) 척도(결합도 척도)가 있는데, 그것들을 유지보수 노력을 예측하는데 이용하여 왔다. 유지보수 작업 중 약 50% 이상의 노력이 프로그램을 이해하는데 드는 것으로 알려져 있기 때문에 프로그램 이해와 관련된 연구들이 많이 행해졌는데, Letovsky와 Soloway[1, 2]는 계획이 변경될 때 발생하는 프로그램 이해의 어려움을 증명하였고, 프로그램 내부 구조 정도와 프로그램 표현 형태 또한 프로그램 이해에서 중요한 역할을 한다고 지적하였다. 그리하여 프로그램 크기와 관련된 클래스의 수, 메소드의 수, 메시지의 수, 프로그램 라인 수 등이 복잡도에 영향을 주는 요인들로 제시되고 측정되었다.

한편 인지 과학을 소프트웨어 공학에 접목시킨 연구들이 많이 행해져왔다. Miller[3]는 실험을 통해 실체를 처리할 수 있는 인간의 단기 기억(short-term memory) 용량이 7±2 chunks임을 알아내었다. 유사하게 프로그래밍 경험은 제어 논리를 취급하기 위해서는 적은 능력을 가진다고 제시하였다. 프로그램에서 종속된 제어 구조의 수는 3 주위로 한계된다고 하였다. 한편, 경험은 프로그램 제어 논리를 처리할 때 또한 3±1 논리 제어구조에 한계된다는 것을 제시하였다[4]. Gugerty[5]등과 Littman[6]등은 인지 과학 측면에서 프로그램 이해와 관련된 연구를 하였는데, 여기에서 인간의 단기 기억이 프로그램 이해에 중요한 역할을 하고 있다고 설명하고 있다. 인지 심리 이론에 근거를 둔 지침들은 프로그램 구성성분들은 실세계를 표현한 청크(chunk)이므로, 소프트웨어 개발이나 복잡도 측정에 있어서 인간의 단기 기억의 한계 7±2를 고려하여야 하며, 특히 복잡도 측정에서는 소프트웨어 규모, 복잡도 그리고 난이도 등에 영향을 주는 요인 수 증가에 따라 가중되는 인지 심리 복잡도를 반영해야 한다고 주장하고 있다.

따라서, 본 논문에서는 프로그램의 볼륨과 복잡도의 관계에 대하여 통계적 분석을 수행하고 인지 심리 이론과의 관계를 파악해 보고자 한다.

2. 프로그램 분석

우리는 다음 표 1에 설명된 13개의 척도를 이용하여 Java 프로그램의 성질을 분석하였다[7,8,9,10]. 척도 값 추출에는 썬 마이크로시스템즈사의 JDK 1.2를 구성하는 awt 패키지의 프로그램 220개를 사용하였다. 척도 값의 측정은 Banda의 Java Source Metrics[11]와 Andrew와 Rajesh의 JMetric[12] 등의 자동화 도구를 주로 사용하였고, 일부 측정 프로그램은 C++로 구현하였다.

표 1 측정 소프트웨어 척도

척도	설 명
WMC	클래스내 메소드의 사이클로매틱 복잡도 (Cyclomatic Complexity)값의 합
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
DIT	클래스 상속 깊이
NIM	호출된 메소드(Invoked Method)의 수
NOCB	메소드에서 참조하는 협력자(Collaborator)의 수
LCOM	클래스의 응집도의 결핍
Coh	클래스의 응집도
NIC	내부 클래스(Inner Class)의 수
DOIC	내부 클래스(Inner Class)의 상속 깊이

3. 인자 분석(Factor Analysis)

자료 집합의 변수 그룹이 강하게 상관관계가 있다면 이들 변수는 측정할 객체의 같은 근원적인 차원을 측정할 것이다. 인자 분석은 변수들간의 상관관계를 고려해서 이들 측정치사이에 공유하는 구조를 파악해 내는 기법이다. 이를 통해 연구자에게 주어진 많은 정보를 분석이 용이하도록 적은 수의 인자(factor)로 제시해 주는 분석 방법이다[13].

인자 분석을 수행한 결과 표 2에서 보는 것처럼 첫 번째 인자에는 크기 척도인 WMC, NOM, NOPM, LOC, SLOC와 호출 관계를 나타내는 결합도 척도인 NIM, NOCB가 포함되어 프로그램의 볼륨을 표현하고 있다. 우리는 이 첫 번째 인자에 포함된 크기/결합도 척도와 인자 분석에서 발생된 인자 점수와의 관계 분석으로 인해 척도 값의 변경이 인자에 미치는 영향을 알 수 있다.

그림 1~7에서 보는 것처럼 인자에 속한 척도들의 값이 커질수록 인자점수가 커지고 있지만 일정 수준에 도달한 후부터는 증가가 둔화되는 것을 알 수 있다. 데이터 분석에 따르면 LOC의 경우 75~100이 넘으면서 증가가 둔화되고 있고, SLOC의 경우 40~55를 넘을 경우 증가가 둔화되고 있다. NOM경우 15~20이 넘으면서 증가가 둔화되고 있고, NOPM의 경우 10~15를 넘을 경우 증가가 둔화되고 있다. NOCB의 경우 8~13이 넘으면서 증가가 둔화되고 있고, NIM의 경우 21~27을 넘을 경우 증가가 둔화되고 있다. WMC의 경우 26~34를 넘을 경우 증가가 둔화되고 있다.

표 2 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 var.	44.255	14.898	12.120	8.253	7.813

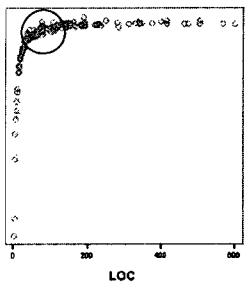

그림 1 LOC와 인자와의 관계

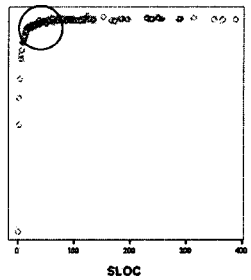

그림 2 SLOC와 인자와의 관계

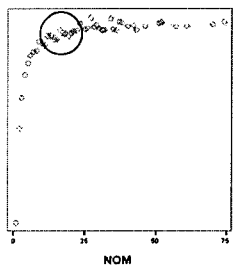

그림 3 NOM과 인자와의 관계

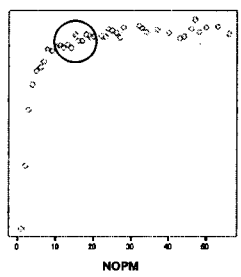

그림 4 NOPM과 인자와의 관계

이것은 인지 심리 이론에서 주장하고 있는 것처럼 인간이 이해할 수 있는 한계내에서는 급격한 복잡도의 증가가 발생하지만 한계를 벗어났을 경우에는 그 복잡도를 측정할 수 없고 그 의미가 없어진다는 것과 관계가 있다. LOC와 SLOC의 경우를 보면 인지 심리 이론에서 인간 단기 기억의 한계로 주장하고 있는 7±2 값에 근접한 값을 취하는 것을 알 수 있다. 한 클래스의 라인수는 메

소드 7±2개가 7±2 라인 수를 가질 경우에 약 50에서 81의 값을 가지는 것을 알 수 있다. 클래스내 협력자의 수인 NOCB 척도 값도 7±2에 근접한 값을 취하고 있다. 호출 메소드의 수를 계산하는 NIM 척도와 클래스내 메소드의 사이클로매틱 복잡도 합을 계산하는 WMC 척도는 제어 논리와 관계가 있다. 인지 심리 이론에서 제어 논리는 3±1로 한계된다고 하였기 때문에 각 클래스에 포함될 메소드 7±2개가 3±1개의 제어 논리를 포함했을 경우 약 21~36의 값을 가지는 것을 알 수 있다.

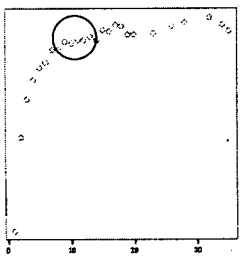

그림 5 NOCB와 인자와의 관계

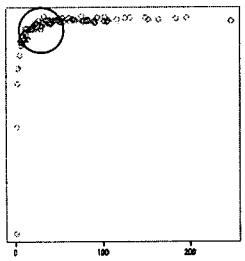

그림 6 NIM과 인자와의 관계

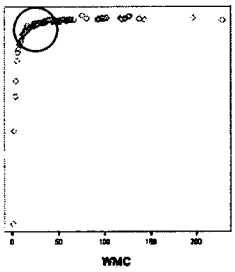

그림 7 WMC와 인자와의 관계

4. 결론

프로그램 볼륨과 복잡도와의 관계를 통계적으로 분석하였다. 프로그램 볼륨과 관계된 척도와 인자와의 관계를 분석하였을 때 프로그램 볼륨이 인자에 미치는 영향은 인지 심리 이론에서 주장하는 인간의 단기 기억 한계 7±2와 제어 논리 한계 3±1 값에 근접한 값을 취한다는 것을 알 수 있다. 이것으로 향후 복잡도 척도를 제안할 때 프로그램 볼륨은 인간의 단기 기억 한계와 제어 논리 한계를 고려하여 측정해야 한다는 것을 알 수 있다.

5. 참고문헌

[1] Letovsky S., "Cognitive Processes in Program Comprehension," *Proceedings of the First Workshop on Empirical Studies of Programmers*, 1986.

[2] Letovsky S. and Soloway E., "Strategies for Documenting Delocalized Plans," *IEEE Software*, pp. 41-49, May 1986.

[3] G. A. Miller, "The Magical Number Seven Plus or Minus Two : Some Limits on Out Capacity to Process Information," *Psychological Reviews*, Vol. 63, pp. 81-87, 1956.

[4] Franck XIA, "Module Coupling: A Design Metric," APSEC '96, Dec. 1996.

[5] Gugerty L. and olson G., "Comprehension differences in debugging by skilled and novice programmers," *Proceedings of the First Workshop on Empirical Studies of Programmers*, 1986.

[6] Littman D. C., Pinto J., Letovsky S. and Soloway E., "Mental Models and Software Maintenance," in *Proceedings of the Second Workshop on Empirical Studies of Programmers*, 1987.

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

[8] 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, June 1994.

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