OCLC FirstSearch: Display

Your requested information from your library TENNESSEE TECH UNIV

Return

SHIPPED - Lender

53627625

General Record Information

Request Identifier: 53627625
Status: SHIPPED
Request Date: 20090429
Source: FSILLSTF
OCLC Number: 4583109
Borrower: TMS
Need Before: 20090512
Receive Date:
Renewal Request:
Due Date: N/A
New Due Date:
Lenders: *TTU, TUC, NHI, MWR, FWA
Request Type: Copy

Bibliographic Information

Call Number:
Title: The Journal of systems and software.
ISSN: 0164-1212
Imprint: [New York, Elsevier North Holland] 1979 9999
Article: Ho-Won Jung Marjan Pivka Jong-Yoon Kim , "An empirical study of complexity metrics in Cobol programs"
Volume: 51
Number: 2
Date: April 2000
Pages: 111-118
Verified: WorldCat Availability: Alternate: Subscription Fulfillment, Elsevier North Holland, 52 Vanderbilt Ave., New York, NY 10017 CODEN: JSSODM Desc: v. Type: Serial, Internet Resource

My Library's Holdings Information

LHR Summary: 7-(1987-)
Lending Policies: Unknown / Unknown
Location: TTUU
Copy: 1
Format: unspecified

Borrowing Information

Patron: Korson, Tim
ARIEL: lib-ariel.southern.edu ILL/McKee Library/Southern Adventist Univ./PO Box

ELSEVIER

The Journal of Systems and Software 51 (2000) 111–118

The Journal of Systems and Software

www.elsevier.com/locate/jss

An empirical study of complexity metrics in Cobol programs

Ho-Won Jung [a] *, Marjan Pivka [b], Jong-Yoon Kim [c,1]

[a] Department of Business Administration, Korea University, Anam-dong 5Ka, Sungbuk-gu, Seoul 136-701, South Korea
[b] The Faculty of Business Economics, Razlagova 14, P.O. Box 180, 62000 Maribor, Slovenia
[c] National Information & Credit Evaluation Inc., Ah-Tae Building, 1337-20, Seocho-2dong, Seocho-Ku, Seoul 137-751, South Korea

Received 4 December 1998; received in revised form 24 March 1999; accepted 13 June 1999

Abstract

Complexity has been used as an important means for exploiting codes to predict or improve program quality and to measure the impact of maintenance costs. Many of the complexity metrics have been known to measure a similar construct of program codes. In revealing the underlying construct of 13 complexity metrics, this study divides the 368 Cobol programs into four classes with respect to the lines of code and then performs factor analysis on optimally scaled complexity metrics. In each factor, a representative metric is identified and then used to predict the remaining metrics. The results show that each class has a slightly different set of metrics in factors and that the representative metric can successfully estimate the remaining metrics in the same factor. The findings indicate high quality estimations in the 12 Halstead software-science metrics by utilizing a McCabe cyclomatic metric at the end of the design phase. In addition, the lines of code are nicely fitted by utilizing each complexity metric as an independent variable. The fitted lines are computed by two methods; least squares (LS) and relative least squares (RLS). The predictive quality of the LS and RLS is revealed to be dependent on the metrics.

1. Introduction

Complexity refers to "the characteristics of the data structures and procedures within the software (program code) that make it difficult to understand change" (Curtis et al., 1979). Complexity has been used as an important means for exploiting program codes to predict or improve program quality (Li and Cheung, 1987; Khoshgoftaar et al., 1992a,b; Hops and Sherif, 1995). It has also been shown to be a major factor in influencing the maintenance effort (Banker et al., 1998; Kemerer, 1995). Many studies revealed that "many of the metrics measure essentially the same construct" (Coupal and Robillard, 1990; Munson and Khoshgoftaar, 1993; Mata-Toledo and Gustafson, 1992). The same situation applies to complexity metrics.

Since many complexity metrics measure a similar construct, some metrics are highly intercorrelated, i.e., a high degree of multicollinearity. The multicollinearity creates sensitivity problems of the estimated coefficients due to small changes, addition, or deletion of independent variables in the regression analysis (Khoshgoftaar and Munson, 1990). This problem of multicollinearity questions the predictive quality of the regression model.

With many complexity metrics having characteristics of multicollinearity, this study utilizes factor analysis to identify the underlying dimensions in 13 complexity metrics from 368 Cobol programs. The 368 Cobol programs are divided into four classes with respect to lines of code. Then, we perform factor analysis on some transformed metrics followed by an optimal scaling in order to fit factor analysis in each class.

In each factor, a representative metric is defined as the metric with the largest factor loading. The representative metric is used to predict the remaining metrics within the same factor. The concept of the representative metric can reduce costs by collecting and keeping all possible valuable metrics. This study utilizes two regression methods such as least squares (LS) and relative least squares (RLS) to drive the relationships between metrics.

Furthermore, this study shows that a McCabe metric (McCabe, 1976), available in the design phase, can predict Halstead-science metrics (Halstead, 1977) computed in the coding phase. Lastly, we show a method to escape the multicollinearity in the prediction of code size with complexity metrics.

* Corresponding author. Tel.: +82-2-3290-1938; fax: +82-2-922-7220.
E-mail addresses: hwjung@kuccnx.korea.ac.kr (H.-W. Jung), pivka@uni-mb.si (M. Pivka), kjyljy@nice.co.kr (J.-Y. Kim).
[1] Tel.: +82-2-3475-5832; fax: +82-2-3475-5500

0164-1212/00/$ - see front matter
PII: S0164-1212(99)00116-8

2. Metric collection and basic statistics

2.1. Metric collection

We have analyzed 13 complexity metrics from the 368 Cobol programs collected from Slovenian industry. Although there are many complexity metrics, this study focuses only on a McCabe cyclomatic metric and 12 Halstead metrics. The Cobol programs have a range of code sizes from 40 to 3722 excluding blank and comment lines. The metric data have been collected by an automatic tool (proprietary), SOME (SOftware MEtric) written by MODULA 2, developed at the Faculty of Business Economics in Maribor, Slovenia. The notations and their explanation are as follows:

Notation

1.	n_1	number of unique operators
2.	n_2	number of unique operands
3.	N_1	total number of occurrence of operators
4.	N_2	total number of occurrence of operands
5.	n	program vocabulary: $n_1 + n_2$
6.	N	program length: $N_1 + N_2$
7.	V	program volume: $N \log_2(n_1 + n_2)$
8.	EN	estimated program length: $n_1 \log_2(n_1) + n_2 \log_2(n_2)$
9.	E	effort: $V/[(2/n_1) * (n_2/N_2)]$
10.	PL	program level: 1/DL
11.	DL	program difficulty level: $(n_1/2)/(N_2/n_2)$
12.	L	language level
13.	VG	McCabe cyclomatic number: number of decisions + 1
14.	LOC	lines of code (excluding blank and comment lines)

The first 12 metrics (1–12) belonging to the Halstead software-science represent software complexity. The Halstead metrics measure the properties of a program to predict program length, volume, difficulty level, program level, and others. Since the metrics are available after the coding phase, they are utilized in the testing and maintenance phases. The metric (13), a McCabe cyclomatic number, measures the control flow structure in order to predict the difficulty of program understanding and the extent of program defects. The McCabe metric is available after the detailed design phase. The last metric (14) is the size of the program excluding the comment and blank lines.

2.2. Basic statistics

The second column of Table 1 shows the mean and standard deviations of the 368 Cobol programs, where the metric LOC has a high standard deviation of 494.18 and a mean value of 451.15 with minimum and maximum values of 40 and 3722, respectively. This study divides the 368 programs into four classes based on the LOC size in order to investigate the size effects and to stabilize the data set. The sample size of Classes 1, 2, 3, and 4 are 92, 93, 92, and 91 programs, respectively. Since the last two programs of Class 2 have the same LOC size, Class 2 has the sample size of 93. The analysis for the four classes is only applied to factor analysis and its related regression.

This study analyzed the correlation matrices of the 14 metrics for the four classes to investigate the characteristics of the data set (the correlation matrices are omitted). From the correlation matrix in each class and Table 1, points of interest are summarized as follows:

- For all metrics, the standard deviation is smaller than the mean.
- In Classes 2, 3, and 4, the means of all metrics except PL and L have increased compared to the previous class, whereas the means of the metric PL is decreased in Classes 2, 3, and 4. Note that the metric PL has an inverse relationship with DL. The metric L has a pattern of increased, decreased, and increased in Classes 2, 3, and 4, respectively.
- The standard deviations of all metrics except n_1 and L have the same increase or decrease patterns as the means.
- High correlation between many metrics.
- Since n and N have high correlation with other metrics, they are omitted in factor analysis to escape singularity problems. In regression, the two metrics are estimated with V and EN, respectively.
- PL has negative or zero correlations with the remaining metrics in all classes.
- As noted in other studies (Basili and Hutchins, 1983; Hops and Sherif, 1995; Ramamurthy and Melton, 1988), N and V are highly correlated with LOC (the coefficients of 0.98 for the both cases).
- As noted by Shen et al. (1985, p. 321), for PL/S (system derivative of PL/1) and assembly language, the McCabe metric VG has the highest correlation with n_2.

3. Factor analysis

3.1. Description of factor analysis

The purpose of factor analysis is to describe the covariance relationships among complexity metrics (variables) in terms of a few underlying constructs. The underlying construct, known as a factor, is a variable or construct that is not directly observable but inferred from input variables. The basic rationale of factor analysis is the creation of highly correlated variables, i.e., high correlations of all variables within the factor

Table 1
Mean and standard deviation[a]

Metrics		Total	Class 1	Class 2	Class 3	Class 4
1.	n_1	41.73	31.03	38.29	45.45	52.26
		(12.26)	(6.74)	(8.25)	(11.55)	(10.32)
2.	n_2	190.01	79.14	141.49	188.79	351.14
		(119.95)	(24.43)	(29.00)	(50.88)	(113.75)
3.	N_1	623.10	168.01	338.31	557.02	1431.47
		(642.71)	(59.59)	(80.58)	(135.59)	(824.83)
4.	N_2	649.88	172.80	350.35	560.37	1518.29
		(690.79)	(64.94)	(98.52)	(125.41)	(895.92)
5.	n	231.75	110.17	179.78	234.24	404.67
		(128.74)	(28.02)	(32.34)	(57.97)	(119.14)
6.	N	1270.28	340.81	688.66	1106.41	2949.76
		(1326.11)	(108.98)	(151.37)	(256.23)	(1701.35)
7.	V	10534.20	2330.49	5170.24	8787.95	25887.46
		(12370.03)	(839.21)	(1283.20)	(2239.89)	(16461.38)
8.	EN	1713.36	658.75	1217.56	1689.51	3292.77
		(1187.48)	(215.72)	(271.94)	(505.34)	(1189.47)
9.	E	1075.09	82.97	252.24	621.48	3347.67
		(2305.78)	(49.00)	(128.77)	(288.91)	(3771.71)
10.	PL	2013.65	3199.52	2249.89	1559.08	1038.60
		(994.55)	(844.30)	(576.71)	(425.07)	(338.72)
11.	DL	63.94	33.06	46.94	68.24	107.76
		(36.76)	(9.47)	(12.85)	(17.94)	(40.69)
12.	L	2311.02	2173.49	2627.87	2129.97	2307.33
		(971.48)	(698.72)	(1240.80)	(974.86)	(816.76)
13.	VG	186.70	55.86	97.81	158.60	435.19
		(224.65)	(20.02)	(26.38)	(50.89)	(333.58)
14.	LOC	451.15	117.72	233.98	386.18	1068.34
		(494.18)	(34.75)	(36.65)	(70.56)	(653.84)

[a] Note that the value in parenthesis is the standard deviation.

but relatively weak correlations with variables of a different factor. Applying factor analysis to complexity metrics creates a set of highly related variables to reduce the number of complexity dimensions.

The factor analysis model can be addressed as follows: assume that p random variables $X = (X_1, X_2, \ldots, X_p)$ of a multivariate distribution have a mean $\mu = (\mu_1, \mu_2, \ldots, \mu_p)$. The random variable X_i can be expressed by a linear combination of m unmeasurable common factors (random variables) and error terms, where $m < p$. A mathematical expression of the factor model is

$$X_i - \mu_i = \sum_{j=1}^{m} \lambda_{ij} F_j + \epsilon_i, \quad i = 1, \ldots, p,$$

where factor loading λ_{ij}, a correlation between X_i and F_j under standardized input variables, denotes the importance of factor j for the variable X_i. Communality $(\sum_{j=1}^{m} \lambda_{ij}^2)$ means the portion of a variable's variance that can be explained by m factors. The covariance between F and ϵ is assumed to be zero.

3.2. Appropriateness of factor analysis

Factor analysis was performed with a varimax for each class. However in some classes, the results did not show the data appropriateness for factor analysis. An optimal scaling with a SAS PRINQUAL procedure was separately performed for each class (SAS, 1994). The optimal scaling with the SAS PRINQUAL procedure is a preprocessing technique that transforms variables to maximize the relations between the observations and data analysis model.

The SAS PRINQUAL procedure recommends the metric transformation for factor analysis as seen in Table 2. All metrics except VG in Class 1 are recommended for a concave transformation. The concave recommendations in Classes 1 and 4 are distinct while the recommendations in Classes 2 and 3 show some ambiguity (the plotted graphs are omitted). Table 2 shows the best transformation among our trials. The metric with a concave recommendation is transformed with a log function.

The data appropriateness of factor analysis was examined with the transformed data set for the four classes. Although there are several examination methods, we followed the three recommendations proposed by Sharma (1996). The first recommendation is to examine the correlation of the metric data. High correlation among metrics indicates that the metrics can be grouped into homogeneous sets of metrics such that each group measures the same underlying constructs. The correlation metric for each class demonstrates the appropriateness of factor analysis for our data (the correlation metrics are not shown).

Table 2
Recommended transformation with optimal scaling

Metrics		Class 1	Class 2	Class 3	Class 4
1.	n_1	Concave			
2.	n_2	Concave			
3.	N_1	Concave		Concave	
4.	N_2	Concave			Concave
5.	n	Concave			
6.	N	Concave			Concave
7.	V	Concave	Concave		
8.	EN	Concave	Concave		Concave
9.	E	Concave		Concave	Concave
10.	PL	Concave	Concave	Concave	
11.	DL	Concave	Concave	Concave	
12.	L	Concave			
13.	VG	Concave	Concave	Concave	Concave

The next recommendation is an examination of partial correlations controlling for all other variables. Since the partial correlations should be small for the correlation matrix, we identified the small partial correlations in order to use factor analysis for all classes (table is omitted).

The third recommendation is the KMO (Kaiser–Meyer–Olkin) measure of individual and overall sampling adequacy. The KMO measure shows the extent of the relationship among indicators in a construct. Sharma recommends that the overall KMO measure should be greater than 0.80, however, he also states that a KMO value of 0.60 is tolerable.

The KMO values for all classes are greater than 0.63 as seen in Table 3. Without optimal scaling, the KMO values were 0.67, 0.54, 0.60, and 0.72 for Classes 1, 2, 3, and 4, respectively. Referring to Table 2, the more distinct the transformation function, the larger the KMO value. The percentage of variance explained in the last row of each class is a summary measure indicating how much of the total original variance of all metrics is represented by the factor. For instance in Class 1, the first, second and third factors explain 69.55%, 17.59%, and 9.08% of the total variance of the 13 metrics, respectively.

3.3. Results of factor analysis

Factor analysis was applied to classify the 13 complexity metrics into a smaller number of factors. The number of factors was determined with a scree plot and with the eigenvalue-greater-than-one-rule.

Table 3 shows the results of factor analyses for the four classes. The cut-off value of 0.6 recommended by Sharma (1996) was satisfied for all of the factors except the value of 0.65 in the first metric of Class 2. Table 3 shows slight differences in the set of metrics in each factor for the four classes. All metrics except n_1, n_2, EN, and VG belong to the same factor for all classes, i.e., n_1, n_2, EN, and VG are sensitive to the LOC size. The

Table 3
Results of factor analysis

Class 1	Factor 1	Factor 2	Factor 3	KMO value
	n_1, n_2, N_1, N_2, V^a, EN, E, PL, DL	L^a	VG[a]	0.76
Variance explained (%)	69.55	17.59	9.08	
Class 2				
	n_1, N_1, N_2, V, EN, E,[a] PL, DL	n_2, L^a	VG[a]	0.65
Variance explained (%)	58.82	24.35	10.96	
Class 3				
	n_1, N_1, N_2, V, E^a, PL, DL	n_2, EN, L^a	VG[a]	0.63
Variance explained (%)	53.81	27.34	14.17	
Class 4				
	n_2, N_1, N_2, V, EN, E^a, PL, DL, VG	n_1, L^a		0.80
Variance explained (%)	73.32	15.20		

[a] Denotes a representative metric in the corresponding factor.

McCabe metric VG is known to be insensitive to program size (Hops and Sherif, 1995), but our result shows that the metric has a different pattern in the LOC size, i.e., the metric is a single element of the third factor in Classes 1, 2, and 3 but grouped with other metrics in Class 4.

Various studies have found that the metrics N_1 and N_2 in a factor were in the same set of other programming languages. The results of a study by Mata-Toledo and Gustafson (1992) revealed that N_1 and N_2 belonged to the same factor in the analysis of 404 Pascal programs. In factor analysis of 240 Ada programs for a data structure complexity, Munson and Khoshgoftaar (1993) showed that N_1 and N_2 were also grouped in the same factor. The same grouping was reported in PL/M programs, where all 10 complexity metrics, identical to our metrics (1)–(9) and (13), were grouped in one factor (Hops and Sherif, 1995). Thus, the metric N_1 and N_2 can be inferred as being insensitive to programming languages. However, the metrics n_1 and n_2 showed sensitivity to programming languages in the above studies for Pascal, Ada, and PL/M as well as Cobol.

The differences of the metric set of a factor can be explained in several points. The kind and number of analyzed metrics are the source of a different set of metrics in factors. These differences may also result from measurement tools (Munson, 1995, p. 273). Since there are no measurement standards for software measurement, the measurement results may differ slightly based on the tool used. In addition, Munson (1995) pointed out the possibility of differences in the enumeration of Halstead n_1 in the analysis of C programs due to operator overloading. For programming styles and standards in Cobol programs written by a trained programmer, all functional program variables tend to be tightly controlled. A single variable will always be used in the same context, i.e., there will be little operand overloading, whereas, programs written by Fortran or C have operand overloading (Munson, 1995).

4. Regression analysis

4.1. Description of regression analysis

Regression analysis is a well-known statistical method that identifies the relation between two or more quantitative variables where a variable can be predicted from others. The most popular method for finding the fitted line is the LS such that the sum of the squared differences between an observed value y_i of dependent variable and the predicted value $\hat{y}_i$ for $i = 1, \ldots, n$ is minimized, i.e.,

$$\min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where the fitted line is considered any number of the independent variables k. Thus, a multiple regression can be written as

$$\hat{y} = b_0 + b_1 x_1 + \cdots + b_k x_k. \tag{1}$$

The observations of software metrics show very large variations for programs or modules (Khoshgoftaar et al., 1992a,b; Mata-Toledo and Gustafson, 1992). For example, a module may have high complexity with a small number of source lines while others may have low complexity with a very large number of source lines. Therefore, certain cases show that the better fitted line can be found by the RLS (Khoshgoftaar et al., 1992a)

$$\begin{aligned}
&\min \sum_{i=1}^{n} \left[\frac{y_i - \hat{y}_i}{y_i} \right]^2 \\
&= \min \sum_{i=1}^{n} \left[1 - \frac{\hat{y}_i}{y_i} \right]^2 \\
&= \min \sum_{i=1}^{n} \left[1 - \frac{b_0 + b_1 x_{1i} + \cdots + b_k x_{ki}}{y_i} \right]^2 \\
&= \min \sum_{i=1}^{n} \left[1 - \left(b_0 \left(\frac{1}{y_i} \right) + b_1 \left(\frac{x_{1i}}{y_i} \right) + \cdots + b_k \left(\frac{x_{ki}}{y_i} \right) \right) \right]^2.
\end{aligned} \tag{2}$$

By letting $y_i^* = 1$, $x_{1i}^* = 1/y_i$, $x_{2i}^* = x_{1i}/y_i$, $\ldots$, $x_{(k+1)i}^* = x_{ki}/y_i$, (2) becomes

$$\min \sum_{i=1}^{n} (y_i^* - \hat{y}_i^*)^2.$$

Then, the fitted line becomes a linear model without an intercept as follows:

$$\hat{y}^* = b_0 x_1^* + b_1 x_2^* + \cdots + b_k x_{k+1}^*. \tag{3}$$

The determination of predictive quality of fitted lines is based on several criteria such as the coefficient of determination R^2, its adjusted R^2, C_p statistic, loss function, and others. This study utilizes the adjusted R^2 and loss function for the decision criteria. The loss function has been employed to determine the quality of the fitted line (Khoshgoftaar et al., 1992a,b). The loss function, i.e., the average of absolute relative error (ARE), is

$$\mathrm{ARE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|.$$

4.2. Regression with a representative metric

Since metrics in the same factor are highly correlated, any one of the metrics can be equally utilized for predicting the remaining metrics. Based on the idea of having a metric with the most impact on a factor, a representative metric is defined as a metric with the largest value of factor loading in a factor. Table 3 shows

the representative metric with a superscript a in each factor.

Table 4 shows the coefficients in the RLS regression of the remaining metrics with the representative ('independent') metric in each factor. The RLS regressions are always superior to the LS in all cases. All estimations of the fitted lines are enough to show the high quality of the fitted lines. Adj R^2 are over 0.95 except the metric VG estimation with E in Class 4, where Adj R^2 is a value of 0.78. Hence, the values of AREs and Adj R^2 are omitted. Note that the coefficients for the metrics n and N are estimated with EN and V, respectively.

4.3. Estimation of Halstead metrics with a McCabe metric

Once the McCabe metric is available after the detailed design phase, it can be utilized to predict the Halstead metrics. Shen et al. (1985, p. 318) cited that "approximately 60% of a group of well-disciplined programmers were able to estimate n_2 after a program design within 25% of final values in two experiments".

This study used the LS and RLS to estimate the relations without dividing classes. Table 5 shows the estimation of Halstead metrics with the metric VG in the design phase. The ARE value of the RLS is consistently higher than ones of the LS except E, while Adj R^2 of the LS is smaller than one of the RLS except E. However, since complexity metrics n_1, n_2, n, EN, PL, DL and L have relative small values of Adj R^2, the LS cannot be recommended to estimate them.

4.4. Estimation of LOC with each complexity metric

This study finds the fitted line of LOC with each complexity metric as an independent variable without dividing classes. As seen in Table 6, all metrics except L show a satisfactory value of Adj R^2 or ARE in the estimation of LOC in the RLS. The best estimation of LOC is from the metrics N and V with Adj R^2s of 0.982 and 0.981, respectively, while by our 13 complexity metrics with an Adj R^2 of 0.985. This result implies that the metrics N or V are enough to estimate the metric LOC while escaping the multicollinearity in the predic-

Table 4
Estimation of metric with its representative metric [a]

Dependent variable		Class 1		Class 2		Class 3		Class 4	
		b_0	b_1	b_0	b_1	b_0	b_1	b_0	b_1
1.	n_1	0.14	0.26	−19.04	10.16	27.56	0.02	63.21	−0.01
2.	n_2	−0.87	0.68	−28.28	20.55	2.99	0.29	−252.39	73.74
3.	N_1	−0.16	0.67	−47.69	64.79	267.63	0.44	−3352.05	598.44
4.	N_2	−1.95	0.92	−403.63	136.12	336.41	0.33	2.75	0.58
5.	n	24.76	0.12	35.22	0.12	40.70	0.12	74.07	0.10
6.	N	39.36	0.13	81.63	0.12	157.14	0.12	281.03	0.10
7.	V	(1)		−7192.33	2246.85	4478.76	6.59	5.35	0.60
8.	EN	0.98	0.71	−691.20	337.30	−2525.95	519.47	−11494	1449.53
9.	E	−7.34	1.51	(1)		(1)		(1)	
10.	PL	0.53	12.13	10.55	−0.53	2020.93	−0.88	12.48	−0.56
11.	DL	−0.52	0.52	−83.78	23.81	3.67		−1.11	0.57
12.	L	(2)		(2)		(2)		(2)	
13.	VG	(3)		(3)		(3)		−584.30	109.18

[a] The number in parenthesis denotes a factor.

Table 5
Estimation of Halstead metrics with McCabe metric

Dependent variable		LS: Eq. (1)				RLS: Eq. (3)			
		b_0	b_1	Adj R^2	ARE	b_0	b_1	Adj R^2	ARE
1.	n_1	37.42	0.02	0.18	0.22	31.80	0.03	0.94	0.25
2.	n_2	112.01	0.42	0.61	0.32	53.88	0.50	0.86	0.47
3.	N_1	112.15	2.74	0.92	0.26	3.59	3.09	0.94	0.26
4.	N_2	115.14	2.86	0.87	0.32	−1.13	3.04	0.91	0.52
5.	n	149.43	0.44	0.59	0.28	90.58	0.51	0.88	0.39
6.	N	227.29	5.60	0.90	0.28	1.38	6.20	0.93	0.37
7.	V	823.20	52.01	0.89	0.32	−434.47	50.10	0.90	0.90
8.	EN	936.26	4.16	0.62	0.34	417.62	4.49	0.83	0.54
9.	E	−691.51	9.46	0.85	2.09	−58.87	2.52	0.74	1.16
10.	PL	2479.43	−2.50	0.32	0.37	1609.46	−0.97	0.85	0.51
11.	DL	38.47	0.14	0.70	0.25	22.86	0.18	0.93	0.28
12.	L	2389.41	−0.42	0.01	0.32	1662.93	−0.09	0.84	0.50

Table 6
Estimation of LOC with complexity metrics

Independent variable		LS: Eq. (1)				RLS: Eq. (3)			
		b_0	b_1	Adj R^2	ARE	b_0	b_1	Adj R^2	ARE
1.	n_1	−429.59	21.10	0.27	0.61	−145.16	8.95	0.77	1.01
2.	n_2	−255.76	3.72	0.82	1.68	33.07	1.96	0.91	0.29
3.	N_1	−17.27	0.75	0.96	0.19	5.75	0.66	0.97	0.16
4.	N_2	−6.10	0.70	0.97	0.11	9.86	0.63	0.96	0.12
5.	n	−341.65	3.42	0.79	2.71	−69.89	1.75	0.91	0.31
6.	N	−17.16	0.37	0.97	0.12	2.68	0.34	0.98	0.11
7.	V	35.79	0.04	0.97	0.10	22.52	0.04	0.98	0.11
8.	EN	−196.04	0.38	0.82	0.99	−22.45	0.22	0.92	0.28
9.	E	238.31	0.20	0.85	0.30	89.18	0.34	0.90	0.31
10.	PL	1068.50	−0.31	0.38	0.87	369.77	−0.07	0.79	0.89
11.	DL	−287.49	11.55	0.74	1.66	−69.11	6.02	0.88	0.43
12.	L	516.21	−0.03	0.00	0.69	95.23	0.02	0.59	2.52
13.	VG	63.63	2.08	0.89	0.27	3.25	2.11	0.92	0.35

tion of code size. Table 6 also shows that LOC can be successfully also estimated by the McCabe complexity metric in the design phase.

As seen in Table 6, the estimations of LOC with L do not fit nicely with an Adj R^2 of 0.59. Thus, we estimated the matric LOC with L in the four classes as follows: in Class 1, $b_0 = 73.4430$ and $b_1 = 0.0079$ (Adj $R^2 = 0.86$ and ARE $= 0.42$; in Class 2, $b_0 = 227.4286$ and $b_1 = -0.0017$ (Adj $R^2 = 0.98$, ARE $= 0.14$); in Class 3, $b_0 = 354.3479$ and $b_1 = 0.0052$ (Adj $R^2 = 0.97$, ARE $= 0.16$); in Class 4, $b_0 = 811.5737$ and $b_1 = -0.0304$ (Adj $R^2 = 0.87$, ARE $= 0.58$), where the fitted lines are followed from (3). At point of Adj R^2, the quality of the above estimation can be said to be an evidence of the usefulness for the four classes.

In addition, we performed a regression to test the relationship in the Halstead metrics as (Cai, 1998)

$$V = L^{1/3}E^{2/3}. \tag{4}$$

Our fitting with an Adj R^2 of 1 was

$$V = 2.313L^{0.334}E^{0.664}. \tag{5}$$

Comparing (4) and (5) shows that the exponents of L and E in our model are the same as the ones used by Halstead. However, our mode has a coefficient of 2.313. In a model without an intercept, our fitting with a Adj R^2 of 0.999 becomes

$$V = L^{0.612}E^{0.695}.$$

The difference between our fitting and Halstead's may come from the difference in programming languages as noted in Section 3.3.

5. Concluding remarks

Many of the complexity metrics measure essentially the same underlying construct of program codes. This study revealed that the complexity metrics were related to each other in the Cobol programs. In factor analysis, this study divided the 368 Cobol programs into four classes with respect to the lines of code. The results showed that some of our 13 complexity metrics recommended a concave transformation. The transformation depended on the metrics and classes. In addition, the set of metrics in factors was slightly different in the four classes. From factor analysis, we can say that the Cobol program has a different underlying structure depending on the metric LOC size. The fitting quality of the LS and RLS depended on the metrics as a dependent variable. This rationale pervaded for all of our regressions.

Since metrics in the same factor are highly correlated, the representative metric with a value of the highest factor loading can be equally utilized for predicting the remaining metrics. Results showed that the representative metrics were enough to predict the remaining complexity metrics. In addition, this study has shown that the McCabe cyclomatic complexity metric could be used to predict the Halstead complexity metrics by utilizing the LS and RLS in the design phase. In the estimation of the metric LOC with a single complexity metric, the result revealed that the metrics N or V were the best metric to predict the LOC while escaping the multicollinearity problems due to the high correlations of metrics. In addition, we experimented with the Halstead's relations of V, L, and E. The result has shown the same values of exponents, but with a different coefficient. This may be a natural outcome due to the difference in programming languages.

References

Banker, R.D., Davis, G.B., Slaughter, S.A., 1998. Software development practices, software complexity, and software maintenance performance: a field study. Management Sci. 44 (4), 433–450.

Basili, V.R., Hutchins, D.H., 1983. An empirical study of a synthetic complexity family. IEEE Trans. Software Engrg. 9, 664–672.

Cai, K.Y., 1998. On estimating the number of defects remaining in software. J. Syst. Software 40, 93–114.

Coupal, D., Robillard, P.N., 1990. Factor analysis of source code metrics. J. Syst. Software 12, 263–269.

Curtis, B., Sheppard, S.B., Milliman, P., Borst, M.A., Love, T., 1979. Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. IEEE Trans. Software Engrg. SE-5, 96–104.

Halstead, M., 1977. Elements of Software Science. Elsevier, New York.

Hops, J.M., Sherif, J.S., 1995. Development and application of composite complexity models and a relative complexity metric in a software maintenance environment. J. Syst. Software 31, 157–169.

Kemerer, C., 1995. Software complexity and software maintenance: a survey of empirical research. Ann. Software Engrg. 1, 1–22.

Khoshgoftaar, T.M., Munson, J.C., 1990. Predicting software development errors using software complexity metrics. IEEE J. Selected Areas Commun. 8, 253–261.

Khoshgoftaar, T.M., Munson, J.C., Bhattacharya, B.B., Richardson, G.D., 1992a. Predictive modeling techniques of software quality from software measures. IEEE Trans. Software Engrg. 18, 979–987.

Khoshgoftaar, T.M., Bhattacharya, B.B., Richardson, G.D., 1992b. Predicting software error, during development, using nonlinear regression models: a comparative study. IEEE Trans. Reliability 41, 390–395.

Li, H.F., Cheung, W.K., 1987. An empirical study of software metrics. IEEE Trans. Software Engrg. SE-13, 697–708.

McCabe, T.J., 1976. A complexity measure. IEEE Trans. Reliability SE-2 (5), 308–320.

Mata-Toledo, R.A., Gustafson, D.A., 1992. A factor analysis of software complexity measures. J. Syst. Software 17, 267–273.

Munson, J.C., 1995. Software measurement problems and practices. Ann. Software Engrg. 1, 255–285.

Munson, J.C., Khoshgoftaar, T.M., 1993. Measurement of data structure complexity. J. Syst. Software 20, 217–225.

Ramamurthy, B., Melton, A., 1988. A synthesis of software sciences measures and the cyclomatic number. IEEE Trans. Software Engrg. 14, 1116–1121.

SAS/STAT User's Guide, vol. 2, version 6, fourth ed., SAS Institute.

Sharma, S., 1996. Applied Multivariate Techniques. Wiley, New York.

Shen, V.T., Yu, T.-J., Paulsen, L.R., 1985. Identifying error-prone software – an empirical study. IEEE Trans. Software Engrg. SE-11, 317–323.

Ho-Won Jung received his Ph.D. from the Department of Management Information Systems at the University of Arizona in 1990. He received his B.S. degree in 1979 from Korea University and his M.S. degree in Industrial Engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1981. He has held a visiting position at Clemson University and has worked at National Computerization Agency (NCA) in Korea. He is currently an associate professor at the Department of Business Administration, Korea University. He has published in ORSA Journal on Computing, Computer and OR, European Journal of Operational Research, IEEE Software, International Journal of Systems Science, and several international conference proceedings. His interest areas include software process assessment and quality evaluation, performance analysis of computer systems and communications networks, and algorithm design for interior point methods.

Marjan Pivka received his M.S. (1980) and Ph.D. (1991) from the University of Zagreb and PgD from Sheffield Hallam University UK (1998). He is currently an associate professor at the Faculty of Economic and Business at the University of Maribor, Slovenia. He was a visiting professor at the International Center for Promotion of Enterprises at University of Ljubljana. He has published in Informatica, IS Audit and Control Journal, QZ Qualitaet und Zuverlaessigkeit and in several European conference proceedings. His interests are quality systems, testing of software packages, software metrics and TQM models.

Jong-Yoon Kim received his M.S. from the Department of Business Administration at Korea University in 1999. He received his B.S. degree in Statistics from Korea University in 1997. He is working for National Information & Credit Evaluation Inc. His interests include software metrics and statistical applications to software engineering.