Software Size Estimation Model for 4GL Systems

(4GL 시스템에 대한 소프트웨어 크기 추정 모델)

Myoung-Young Yoon(윤 명영)*
*Department of Computer Science, Chungcheong College
(충청대학 컴퓨터학부 부교수)

요약

소프트웨어 프로젝트 관리과정의 활동에서 프로젝트 관리자의 중요한 임무는 소프트웨어의 크기와 인적 노력 등을 추정하는 것이다. 최근 소프트웨어 개발에 주로 사용되는 제 4세대 언어(4GL)와 데이터베이스 환경에서 개발되는 응용시스템에 대한 크기를 예측하는 모델은 불행하게도 연구가 미비한 실정이다. 본 논문에서는 4GL로 개발되는 프로젝트 개발 초기 단계에서 수집한 메트릭스를 이용하여 소프트웨어 크기를 예측하는 추정 모델을 제안 한다. 제안된 방법은 상대오차(MRE)를 최소화시키는 방법으로 개발 초기과정에서 얻어지는 측도들의 이상치에 덜 민감한 특성을 가지고 있다.

본 논문에서 제안된 모델에 대하여 적합도와 예측력의 성능을 테스트하기 위하여 데이터 셀을 I과 II 2개로 나누어 실험하였다. 실험결과, 추정된 모델의 적합도와 예측력은 데이터 셀 I과 II 모두에서 제안된 MRE 추정방법이 전통적인 방법 LS, RLS보다 우수하게 나타났다.

Abstract

An important task for any software project manager is to be able to predict and control project size. Unfortunately, there is comparatively little work that deals with the problem of building estimation methods for software size in fourth-generation languages systems. In this paper, we propose a new estimation method for estimating for software size based on minimum relative error(MRE) criterion. The characteristic of the proposed method is insensitive to the extreme values of the observed measures which can be obtained early in the development life cycle.

In order to verify the performance of the proposed estimation method for software size in terms of both quality of fit and predictive quality, the experiments has been conducted for the dataset I and II, respectively. For the data set I and II, our proposed estimation method was shown to be superior to the traditional method LS and RLS in terms of both the quality of fit and predictive quality when applied to data obtained from actual software development projects.

1. Introduction

For managing software production, it is particularly useful to be able to achieve an accurate estimate of the scale of programs being developed as early as possible. When a program is developed, typically the size is estimated experimentally with reference to similar programs that have been developed previously. Program production control, including development management and quality control, can be made more effective if a way is devised to make these experimental program size estimates more

quantitative and formalized, as well as more accurate. An important task for any software project manager is to be able to predict and control project size from measures which can be obtained relatively early in the development life cycle. Since the estimated software size are used for predicting the cost, development effort in the software development projects.

Recently, there is increasing pressure to develop and quantify measures of software size for fourth-generation languages(4GLs) systems that tend to be utilized for 'data strong' applications. Demarco[1] made a distinction between software development systems that were characterized as being 'function strong' and those that were 'data strong'.

A number of proposals have been made to date on formalized, model based methods for estimating program size. One of the early approaches is that of Chrysler[2], who devised an estimation model based on the influence of various program characteristics on program size. One of the best-known models is the function point (FP) method proposed by Albrecht [3]. The FP method classifies functions in the functional specifications according to predetermined categories, and assigns an FP number based on the level of functional complexity. This FP number is then adjusted by means of productivity determinant factors before being used to calculate the estimated program size. Other researchers have come up with methods similar to the FP approach or have attempted to improve on it[4, 5, 6]. These approaches has concentrated been upon function strong systems or the function aspects of systems, however, so that FP approaches are inadequate for the 4GLs systems. Jeffery *et al.*[7] found that a prediction system based upon function points was able to explain less than 40% of the variation in effort. Although significant effort has been devoted to strengthening the counting practices associated with function points[8], questions of subjectivity and measure interdependence remain. Moreover, function point counting is quite a complex process that requires a degree of training.

Recently, Verner and Tate[9] reported upon their attempts to predict size and effort required for a 4GL implementation. Their solution was to use function points to estimate the size of the application, convert this into lines of code(*LOC*) and then to use this figure as an input to the COCOMO model. The results, especially the size estimation, appear to have been quite accurate although a note of caution is required since Vener and tate were only studying a single system. Borque and Cote[10] describe an empirical study where they attempted to predict the size of 4GL systems based upon various metrics derived from ER diagram. Using linear regression they were able to develop effective prediction systems although they noted the need to calibrate the models to the specific measurement environment. A similar approach was suggested by Ince *et al.*[11] and Gray *et al.*[12] and indeed our data collection includes the raw counts required for the more complex synthetic metrics proposed by these authors. However, the prediction systems implied by their work remain unavailable.

To cope with these difficulties, we propose statistical estimation model for software size in particular 4GLs systems on the basis of the linear regression. The traditional method employed by elementary statistic books and most canned statistical

software is that of least squares estimation. In this case the fitted line is chosen so that the sum of the squared differences between an observed value, y_i , of the dependent variable and the predicted value $\hat{y}_i$, is minimized, i.e., min $\sum_{i=1}^{N}(y_i-\hat{y}_i)^2$. This is called least squares (LS) estimation. In addition to the traditional LS and the relative least squares(RLS) estimation methods, in this paper, we propose a new estimation method for prediction software size based on minimum absolute relative error(MRE) criterion.

2. Regression Analysis

In this section we review the traditional regression techniques and propose the MRE regression with minimum relative error criterion so as to estimate software size In this regression model, the metrics will appear as independent variables and the dependent variable will be program sizes. Linear regression models are formed by choosing the best subset from a set of independent variables in order to explain as much variation in the dependent variable as possible. Coefficients for the independent variables are produced by the traditional method LS, RLS, and the proposed method MRE regression techniques that attempt to fit these variables to sample data. Comparative results between LS and least absolute value(LAV) methods and their individual properties have already been presented in various studies[13,14,15]. Several methods of selecting an appropriate subset of independent variables that will not introduce additional variance (or noise) in the model have already been discussed by McDonell *et al.* [16].

2.1 RLS Regression Analysis

The relative least squares(RLS) method involves minimization of the sum of squared errors relative to the oserved dependent variable values[17]. Moreover, note that the RLS procedure can be formally reduced to a regular LS regression model by taking the following steps:

$$\min\left[\sum_{i=1}^{N}\left(\frac{y_i-\hat{y}_i}{y_i}\right)^2\right]=\min\left[\sum_{i=1}^{N}\left(1-\frac{\hat{y}_i}{y_i}\right)^2\right] \quad (1)$$

where
$y_i=b_0+b_1x_1+b_2x_2+\ldots+b_kx_k$. By letting $y_i^*=1$ and $y_i^*=\frac{\hat{y}_i}{y_i}$, it follows that

$$\begin{aligned} y_i^* &= (b_0+b_1x_1+b_2x_2+\ldots+b_kx_k)/y_i \\ &= b_0\left(\frac{1}{y_i}\right)+b_1\left(\frac{x_{1i}}{y_i}\right)+b_2\left(\frac{x_{2i}}{y_i}\right)+\ldots+b_k\left(\frac{x_{ki}}{y_i}\right) \\ &= b_0x_{1i}^*+b_1x_{2i}^*+b_2x_{3i}^*+\ldots b_kx_{(k+1)i}^* \end{aligned} \quad (2)$$

where $x_{1i}^*=1/y_i$, $x_{2i}^*=x_{1i}/y_i$, $x_{3i}^*=x_{2i}/y_i,\ldots,x_{(k+1)i}^*=x_{ki}/y_i$. Following substitutions, one can calculate min $\left[\sum_{i=1}^{N}(y_i^*-\hat{y}_i^*)^2\right]$ which follows a LS minimization process and can be easily implemented.

2.2 MRE Regression Analysis

It has been found that minimization results of the average absolute error of a fit $y=f(x)$ given by

$$AAE=\frac{1}{N}\sum_{i=1}^{N}|y_i-\hat{y}_i| \quad (3)$$

provide very little information about the fit's accuracy [18]. Levitin [18] also adds that "in order to get a better idea, one needs to relate the size of errors to the values being approximated". Furthermore, it is often useful in estimation procedures to measure the performance of a model in terms of its relative error. The average of absolute relative errors is an appropriate "loss function" for determining the quality of the prediction equation. The measure of average relative error is defined as:

$$ARE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{(y_i-\hat{y}_i)}{y_i}\right| \qquad (4)$$

ARE has been established to be useful as a measure of model performance associated with complexity metrics in order to identify fault-prone software[19]. In the present work, however, ARE is also used as a measure of forecast quality among the various estimation techniques. In this way, uniformity is retained between the different measurements of quality of fit and forecast accuracy of the estimation methods. Once a model has been fitted, the ARE of any estimation method can be easily obtained by computing equation (4). One can derive Minimum Relative Error (MRE) estimators of the parameters of the model by solving:

$$\begin{aligned}&\min\left[\sum_{i=1}^{N}|(y_i-\hat{y}_i)/y_i|\right]=\min\left[\sum_{i=1}^{N}\left|1-\frac{\hat{y}_i}{y_i}\right|\right]\\ &=\min\left[\sum_{i=1}^{N}\left|1-\frac{b_0+b_1x_1+b_2x_2+\ldots b_kx_{ki}}{y_i}\right|\right]\\ &=\min\left[\sum_{i=1}^{N}\left|1-\left(\frac{b_0}{y_i}+\frac{b_1x_{1i}}{y_i}+\frac{b_2x_{2i}}{y_i}+\ldots\right.\right.\right.\\ &\qquad\left.\left.\left.+\frac{b_kx_{ki}}{y_i}\right)\right|\right]\end{aligned}$$

letting $y_i^*=1$ and

$\hat{y}_i^*=b_0x_{1i}^*+b_1x_{2i}^*+\ldots+b_kx_{(k+1)i}^*$ and following substitutions, one can then solve $\min\left[\sum_{i=1}^{N}|y_i^*-\hat{y}_i^*|\right]$ which is equivalent to estimating the parameters of a model using the least absolute value criterion (LAV). The MRE regression technique can be regarded as a weighted version of the LAV method with the weights equal to reciprocals of values observed.

3. Prediction of Software Size with Metrics

3.1 General system Characteristics

The systems were built over a period of five years by groups of senior students in the Department of Information Science at the University of Otago[16]. Every system was built to satisfy the real requirements of an external client, normally a small business or department in a larger organization. Also, Each system dealt with transaction processing, data retrieval and reporting, and file maintenance activities performed by the organization. On system delivery, the client performed an acceptance test and review. All projects satisfied the requirements of both the client and the course administrators. A wide variety of systems was constructed over he period, in total, more than seventy distinct working systems were developed and reviewed.

The systems were all of small to medium size, as illustrated by the following indicators of scale: each system included an average of eleven data entities and sixty attributes; six reports were producted on average by each system. In terms of code product size, the smallest system was comprised of 309GL source statements, the largest contained more than 2600 statements, and the average size was approximately 1100 code statements. One of the positive

features of the sample is the degree of commonality of several process attributes across the set. All but a few of the systems were built by groups of four developers; the same development methodology was employed in every case; all systems were implemented using the same tool, the Cognos 4GL Powerhouse; and the systems were all of the same generic class(transaction-oriented data processing and retrieval systems). This commonality is advantageous in that these factors can be considered as constant in the analysis, a condition not often encountered in software size research. when they vary, factors such as can clearly have an impact on system size. Given that these potential contributors may be treated as constant, the degree of confidence adopted in regard to any size relationships supported by the data will consequently be greater.

3.2 Data Collection

A study by Sallis et. al.[16] will serve as a constructive sample for the modeling process. In this investigation, two product sets included the system documentation and the implemented code were examined in order to collect the appropriate data. Specification size measures were manually collected from each System Proposal. Two of the authors performed this task so as to obtain as correct a data set as possible. Each author undertook the collection task independently, then the two data sets were compared and any discrepancies were identified and resolved. Within each set of documents a number of measures were extracted in order to address the following questions:

● is data model size related to the size of the implemented system?

● is functional model size related to the size of the implemented system?

This approach was based on the assumption that consideration of the two dimensions of data and function could provide adequate, independent indicators of system size. The measures collected were coarse, in line with one of the objectives of the study; that is, to test for the existence of size relationships using high- level, objective, and easily extracted indicators. The measures of data model size collected if this study were therefore:

● the number of entities depicted in the entity relationship diagram(ENT)

● the number of relationships depicted in the entity relationships diagram(RSP)

● the number of attributes associated with the entity relationships diagram (ATB).

The functional model size measures were of a similarly coarse nature:

● the number of menus depicted in the functional decomposition chart(MNU)

● the number of data edit screens depicted in the functional decomposition chart(EDT)

● the number of reports depicted in the functional decomposition chart(RPT)

● the number of non-menu functions depicted in the functional decomposition chart (NMN)

● the total number of functions depicted in the functional decomposition chart(FDS).

The second product set examined was the implemented code itself. All source program files for each system were scanned automatically by a parsing program to extract the following actual code size measure:

● the total number of source statements in the system (SIZ).

This measure excluded blank and

comment lines of code, and counted run-on lines with a continuation indicator as a single statement. The extraction of the measure was verified manually on a random selection of ten programs by one of the authors to ensure consistency and reliability. No counting errors were identified by this check. Of the 74 systems in the total sample four were incomplete, in that full specification documents were not available. Consequently a usable data set of eight specification measures and one implementation measure was collected from seventy systems.

3.3 Experimental Results and Analysis

1) Descriptive Statistics

General descriptive statistics for each of the variables are shown in Table 1.

<Table 1>Descriptive statistics for each measure

Measure	Variable	Mean	Skew	Outliers
Data Model Size	ENT	11.6	0.78	2
	RSP	10.2	0.91	1
	ATB	64.5	0.77	1
Functional Model Size	MNU	5.6	2.12	6
	EDT	12.0	1.22	4
	RPT	6.8	0.75	1
	NMN	18.8	1.06	3
	FDS	24.4	0.96	-
Implemented Code Size	SIZ	1106.0	1.0	20

The descriptive indicators highlight the absence of significant skewing for all but the MNU and EDT variables. The coefficients of skewness is used as a measure of asymmetry. Further analysis using boxplot distributions enabled outliers to be identified, with the outliers shown in the right-most column of Table 1.

Given the small degree of skewing in some of the distributions, correlation tests were performed using the nonparametric Spearman statistics so as to identify any potentially useful linear relationships between the specification size variables and the implementation size measure, as well as among the specification variables themselves.

<Table 2> Spearman correlation coefficients
(All significant at 0.01 level,
* at 0.05 level, and ** at 0.1 level)

	SIZ	ENT	RHP	ATB	MNU	EDT	RPT	NMN
ENT	.4948							
RHP	.4670	.9525						
ATB	.6635	.6643	.6307					
MNU	.3827	.3790	.3741	.3443				
EDT	.6677	.6834	.6382	.6709	.4818			
RPT	.5271	.2058 (*)	.1750 (**)	.3537	.2989	.3631		
NMN	.7171	.5512	.4977	.6219	.5024	.8473	.7800	
FDS	.7227	.5616	.5236	.6123	.6758	.8251	.7422	.9632

The results are shown in Tables 2. Both sets of correlation statistics provided evidence of strong significant relationships between several of the specification size variables and the implementation size measure. In particular, the relationships between the ATB, NMN and FDS specification measures and implementation SIZ were strong ever. when tested with the more conservative Spearman statistic. This suggested that, for the range of system size considered here, potential predictive relationships might have been derivable. Some cross-correlation was also evident among the specification size variables themselves, suggesting that several of the measures may have been assessing the same size characteristic.

2) Regression Modeling

We would now like to explore the application of these estimation techniques to two data sets I and II which are randomly

selected set of 50 observations. First, we will model the relationship between a set of the collected size measures and the actual software size with the data sets. Second, we scrutinize the predictive quality of the model. From a regression modeling standpoint there are some profound differences between two data sets I and II. The presence of outliers in the y values representing program size is obvious in the dataset I, whereas no outliers were identified in the modeling process of the raw data for the dataset II.

There was substantial variation in the implemented code sizes or actual code size measure which is the dependent variable in the regression model. The independent variables selected in the regression models used in the present study were based on statistical analysis conducted by McDonell *et al.*[16]. For the data set I and II, the data model size ATB and the functional model size NMN is the independent variables selected.

To evaluate the fit of a regression model to these data sets, there are two distinct evaluation criteria that must be met. First, the regression model must adequately represent the linear dispersion of the data. This is the quality of fit criterion.

<Table 3> Model quality of fit

Estimation Procedure	ARE value for data set I (including outliers)	ARE value for data set II (Not including outliers)
LS	0.6688	0.6513
RLS	0.3453	0.5586
MRE	0.3043	0.4409

Second, the model must make meaningful future predictions. This is the predictive quality of the model.

We will now examine the estimation procedures for both quality of fit consideration and predictive quality. To assess the quality of fit of the estimation methods in terms of ARE, each of these methods was used to fit a regression line. For the data set I in which outliers were present, the MRE techniques produced the lowest ARE indexes. As indicated in Table 3, the MRE methods appear to give a better quality of fit than LS and RLS method. The LS method had the highest ARE value which is an indication of its vulnerability to extreme data points. For the data set II, which did not contain any outliers, the MRE techniques demonstrated superior performance over LS and RLS.

By observing the ARE values for each technique, the MRE method appears to be a good estimator in terms of fitting the best line to data set I and II for a pair of variables representing different metrics in each dataset. The LS method exhibited inferior performance in each data sets. The predictive quality of each of the models will assess the ability of these models to make future predictions. We will now examine the predictive quality of each of the estimation techniques.

In the first case, the data set I was used to evaluate the quality of fit for different estimation techniques, the data set II was reserved to forecast using previously obtained regression equations, respectively. Then the roles were reversed for data set I and II with the dataset II was used to derive the model and the dataset I to validate it. Following such computations, the ARE performance criterion was then determined for the LS, RLS, and MRE procedures. From Table 4, it can be seen that the ARE results for data set I and II indicate that from a model determined from data set I and II, respectively. MRE method

appear to give a better predictive quality than the other estimation techniques.

<Table 4> Model predictive quality
* predictions of dataset II (not including outliers),
** predictions of dataset I (including outliers)

Estimation Procedure	ARE value* (using models built from dataset I)	ARE value** (using models built from dataset II)
LS	1.6488	0.8013
RLS	1.2233	0.8486
MRE	0.7143	0.4409

4. Conclusions

This work shows that it is possible to predict software size for 4GL system at a fairly early stage in a project using simple and objective counts derived from a functional specification and data model. The objective of this study was not to present a definitive model for the prediction of program size measures, but rather to explore and evaluate various estimation techniques for the creation of linear models of software size. Results from the analysis of the regression and estimation methods, we found that MRE procedures possess good properties from the standpoint of both model quality of fit and predictive quality. Since MRE method do not exhibit the sensitivity to the perturbations of data at the extreme values of the size metrics.

This study showed that it is possible to predict the size of a 4GL implementation from metrics derived from the functional specification and ER model.

Predicting implementation size at such an early stage in a software project is useful for the practitioner since it gives important insights into the effort required to develop the project. Developers can usefully collect simple measures derived from documents available early in the development process, for instance data models and functional specifications.

References

[01] T. Demarco, Controlling Software Projects. Yourdon Inc., New York NY, 1982.

[02] E. Chrysler, "The Impact of Program and Programmer Characteristics on Program Size," Proceedings National Computer Conference, pp.581-587, 1978.

[03] A.J. Albrecht, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," IEEE Trans. Software Eng. vol. 9, no. 6, pp. 639-648, 1983.

[04] M. Itakura, and A. Takayanagi, "A Model for Estimating Program Size and Its Evaluation," 6th International Conference on Software Engineering, pp. 104-109, 1982.

[05] C.R. Symons, Software Sizing and Estimating: MKII FPA, John Wiley&Sons Ltd: Chichester, UK, 1991.

[06] Y. Miyazaki, S. Yamada, and M. Itakura, "The Concept of Stepwise Software Sizing Model and a Method to Develop the Model," Transactions of International Processing Society of Japan, vol. 32, no. 2, pp.140-148, 1991

[07] D.R. Jeffery, G.C. Low, and M. Barnes, "A comparison of Function Point Counting Techniques," IEEE Trans. Soft. Eng., vol9, no.5, pp. 529-532, 1993.

[08] IFPUG, Function Point Counting Practices Manual-Release4.0, Westerville OH, 1994.

[09] J.Verner and G.Tate,"Estimating Size and Effort in Fourth-generation Development," IEEE Software, vol. 5, pp. 15-22, 1988.

[10] P. Bourque and V. Cote, "An Empirical in Software Sizing with Structured Analysis metrics," Journal of Systems and Software," vol. 15, pp. 159-172, 1991.

[11] D.C. Ince, M.J.Shepperd, A. Pengelly, and H. Benwood, "The Metrification of Data Designs," Proc. 3rd Ann. Oregon Workshop on Soft. Metrics, pp. 17-19, 1991.

[12] R.H.M., Gray, B.N.Carey, N.A. McGlymn, and A.D. Pengelly, "Design Metrics for Database," BT Technology Journal vol. 9, no. 4, pp. 69-79, 1991.

[13] G.Bassett and R. Koenker, "Asymptotic Theory of Least Absolute Error Regressions," JASA, vol. 73, pp. 618-622, 1978.

[14] T. Dielman, "A Comparison of Forecasts from Least Absolute Value and Least Squares Regression," Journal of Forecasting, vol. 5, pp. 189-195, 1986.

[15] Y. Dodge, "An Introduction to L_1-norm based, Statistical Data Analysis," J. Computational Statistics & Data Analysis, vol. 5, pp. 239-253, 1987.

[16] S.G. McDonell, M.J. Shepperd, and P.J. Sallis, "Metrics for Database Systems: An Empirical Study," Proc. of the 4th Soft. Metrics Symposium", pp. 99-107, 1997.

[17] T.M. Khoshgftaar, J.C. Munson, B.B. Bhattacharya, and G.D. Richardson, "Predictive Modeling Techniques of Software Quality from Software Measures," IEEE Trans. on Soft. Eng., vol 18, no. 11, pp. 979-987, 1997.

[18] A. Levitin, "The L_1 Criteria in Data Analysis and the Problem of Software Size Estimation," in Proc. 21st Symp. Interface: Computing Science and Statistics, pp. 382-384, 1989.

[19] V. Shen, T.Y.S. Thebaut, and L. Paulsen, "Identifying Error-prone Software -An Emprirical Study," IEEE Trans. on Soft. Eng., vol. 31, pp. 1985.