

# **A Software Development Cost Estimation Model for Higher Level Language Environments**

**By  
Jeanette N. Morgan Peoples**

## **ABSTRACT**

Project managers and software engineers are responsible for providing reasonable estimates of the projected cost of developing software to a specific set of software specifications. Erroneous cost planning can have disastrous consequences for the firms that under-staff or under schedule projects based on inaccurate estimates of the level of effort required to develop software applications for management information systems. One of the greatest challenges to estimating software costs has been the fact that each project is unique and therefore the efforts to create the unique combination of functions, screens, reports, and associated underlying database tables for holding the data are in themselves unique. Software cost models generally rely on such inputs as estimates of lines of source code, delivered sets of instructions, function points and processing complexity or experience levels to produce cost estimates. These models generally produce inaccurate results when used to estimate the cost of software development in current development environments such as those that use higher level languages or component based software development. Research to develop and test a new model for predicting software development cost was developed. The goals of this research were twofold, 1) determine if a basic cost estimation model which uses the structural objects of a set of 4GL software applications can provide acceptable levels of estimation accuracy; and 2) determine if predicting level of effort directly from structural components is more accurate in estimating actual level of effort than expert-based level of effort estimates for 4GL applications.

## **Background**

Software development has been undergoing change since its beginnings several decades ago. What began as simple attempts to speed up the calculations of human mathematical processes into actions that could be performed by computers operating on binary digits of zeroes and ones, has evolved dramatically. Orders of magnitude of change in software development have occurred, but the manner of estimating the cost of software development has not kept pace with the changes in tools being used. Programmers utilize software development tools to create software applications that a human will use to support their business functions and responsibilities. Development of these applications takes time and since time costs money, the issue of cost for developing certain software applications becomes a concern. Managers have scarce resources to assign to competing projects for the development of these information systems applications.

Cost estimation of software development projects has consistently presented a quandary for managers. This is largely because every software application is based upon a unique set of user requirements that uniquely comprise an application that programmers develop to satisfy these user expectations. Yet project approval committees, finance officers, stakeholders, and accountants require some advance notion of how much it will cost to automate the set of business functions into an information system application. Cost models have been developed, tested, applied, and often discarded in favor of experienced, expert opinion estimation. Larger organizations employ cost estimation practices based on published cost models. However, the base parameters for cost are traditionally translated back to a metric reflecting the early form of software development, which was lines of code that were written by a programmer to instruct the machine to perform certain functions or algorithms.

Modern software development environments are better understood as aggregates of forms, reports, tables, and screens, rather than lines of code. Even the more recent function point variables, such as characterize popular software cost estimation models, reflect function, not the physical structure of an application. In research conducted to develop a new approach to cost estimation appropriate to a group of fourth generation language (4GL) software applications a new model, COSTMO-4GL, showed promise for a basic prescriptive approach to software cost estimation model development. The model is based upon the structural components of 4GL software development environments.

### **Cost Model Research**

In the past few decades of information applications' software development, numerous cost estimation models have been put forth, tried, and proved or disproved using a number of different estimation parameters and methodologies. Kemerer [KEMERER87] conducted a well-known study reporting on the relative accuracy of four software cost estimation models in being able to predict the actual costs of 15 completed software development projects. The results indicated that the models require significant calibration, and the study provided insight to the factors affecting modern software development productivity. Similar results were derived from the study by Gaffney [GAFFNEY96] on three different sets of historical software development projects, when testing the validity of different derivations of the Albrecht Function Points cost model.

In general, none of the research efforts applying these models address whether a cost model constructed from structural components of modern software development environments, such as 4GL, provides: 1) acceptable levels of predictive accuracy for estimating 4GL software development efforts, or 2) a direct estimate of the level of effort to build a 4GL software application, without re-estimation based on some determined productivity factor to translate lines of code (LOC) or function points to a level of effort cost expressed in person days.

## Programming Language Levels

Programming languages and the training to use them to create software applications for management information systems have evolved over the past 40 years. First generation languages required the programmer to have intimate knowledge of machine specific operations. Second generation languages (2GL), called “assembly languages” eliminated the need for using binary instructions in 1's and 0's. But these languages still required a very advanced understanding of machine processing logic, specific to the machine's language. During this period, contemporary cost models for estimating the effort expected to develop a particular set of requirements functions did not exist and estimates were based on expert judgment without benefit of a formal model.

Third generation language (3GL) provided significant productivity gains for programmers by pre-coding much of the machine code and assembly language statements into more natural language command statements. Commonly used functions such as DO, END, If...THEN statements and arguments are features of many of these languages. The use of these higher-level commands meant that fewer lines of code had to be written to execute the same function in a 3GL. Vendors closely protected proprietary features used to attract more buyers to their 3GL products promising greater leverage and productivity by use of their languages and software development tools. Differences were great among FORTRAN, COBOL, Pascal, and other 3GLs, making it difficult for programmers to transition back and forth from one 3GL to another.

During this period, of the 1970s-1980s, initial research to develop methods for estimating the cost of software development was devoted to estimating the size of software development projects [PUTNAM80] [BAILEY81] [BOEHM81] [DEMARCO82]. The early cost models and even the most popular software cost estimation models still in wide use today, provide estimates of effort by using lines of code and subjective judgments that measure complexity or function points counts combined with application characteristics. Yet these models are sorely inaccurate. It has been noted that historical data use inappropriate code decomposition factors not relevant to modern software development environments or tools. Yet these same models developed for the 2GL and 3GL software development paradigms have been the basis for cost estimation in most of the current models still in use today [VERNER92] [EJIOGU91].

Klepper asserts “The popular SLOC metric measures productivity as a function of the number of lines of source code in a program relative to the number of programming hours or months required... Unfortunately, SLOC is very much a function of the language used and is deficient as a measuring tool in studies that compare the productivity of two or more dissimilar source languages.” Productivity is defined as a count of how many lines of code can be written to implement a particular function or application. In other words, if a programmer can write fewer lines of code to implement the same functions in an application constructed of a higher-level language, it could be inferred that the programmer has demonstrated an improved rate of productivity. However, it must be clearly asserted that this would only hold true if the programmer wrote the fewer lines of code in less time or at less cost.

## **Structural Makeup of 4GL Software Development Environments**

The advent of fourth generation language (4GL) tools brought software development closer to the way that end users view the end-product: software business applications are merely a collection of screens or data entry forms, reports, tables storing data, and coded modules that execute calculations or algorithmic functions. 4GL is defined as the evolution from 3GL programming languages to command-driven, structural programming toolsets. During the past decade, business and government agencies involved in large-scale software development programs discovered the potential advantages of fourth generation languages (4GL) and associated software development tools for rapidly developing and deploying capabilities with a minimum requirement for creating lines of code [MYERS86] [TINNIRELLO91]. Some of the application needs cited leading to this structural shift to higher level language (HLL) software development tools are:

1. flexibility to survive the evolution of the organization's goals and priorities;
2. ability to respond quickly to non-real time systems requirements, such as for decision support applications and management information systems;
3. ease in building standard user interfaces;
4. applicability to support distributed processing; and
5. ability to more easily cross heterogeneous environmental platforms.

4GL tools are suited to many business and technical domains that use data for business management information systems. The learning curve for developers to create software applications has been proven to be quicker with 4GL tools than with third generation languages [LEVIN97]. Klepper [KLEPPER95] acknowledges that there have been many claims that 4GL provides significant productivity gains for software development, but few of the claims have been measured and compared empirically. Since so much has been written, but not proven, regarding 4GL productivity, research to understand the cost of developing software in 4GL is timely.

### **The Case Study**

Ejiogu observes that software engineers and researchers looking at development of cost models continue to ignore “numerical thinking” in the development of independent variables that are used in those models. He notes that “given the unsatisfactory results obtained from existing models” it would be prudent for researchers to carefully evaluate and consider the common criticisms of current models [EJIOGU91]:

1. lack of intuitive persuasion;
2. incongruous counting rules (what to include...)

3. improper definition of parameters for measure leading to ambiguity and subjectivity of measure;
4. and above all, pathological absence of mathematical measurable space that can also be flexible enough to permit extension to other metrics.

Ejiogu recommends that developers of models rather examine the structural character of software applications. When wanting to know the expected size of a garage built of wood boards, one would measure the length and width of the boards, not the color or texture which would be “meaningless” metrics to the problem space. Ejiogu states that the failure of existing models may be largely because mathematical arguments related to creating descriptive models have been ignored. Klein [KLEIN95] points out that there is a difference between software development written in lines of code using “classical techniques...portions are all integrated into a single program,” and that using application agent building blocks.

The table below describes the different types of cost models that were surveyed and studied, some of the specific implementations, and the applicability to modern development environments’ software cost estimation which are characterized by higher level languages. While many cost models consider the specific characteristics of the desired end-product, they do not address measurement of the components of 4GL software development applications. Given the ramifications to industry for continuing to develop poor estimates of effort for developing information systems further study of the appropriate variables and modeling relationships is both compelling and timely.

TABLE 1: Historical Cost Models’ Applicability to 4GL

<b>Expert-Based Models</b>	Rely upon expert judgment; generally subjective (not empirical) interpretation of the value of independent variables
Delphi Techniques (IBM)	Delphi techniques could give averages of counts of structural objects for a set of requirements.
Albrecht/Arthur Function Points models	use estimated counts of data files and system transactions; require judgment of complexity for each of five factors. Arthur adjustment adds consideration of characteristics of application. Discerning factors in 4GL requires judgment, as not derived from 4GL component structures.
Gaffney Simplified Function Points	found two factors explained similar levels of variance; results held up fairly consistently across three diverse groupings of applications. Used regression and correlation analysis to derive equations; simplified models of fewer cost factors may sometimes be equally good predictors of effort.
Fairley Work Packages	hierarchical scheduling tools could allow the manager to track modular costs of developing forms, reports, tables, modules; historical data could be used to improve estimation accuracy.

<b>Static Linear and Non-Linear Models</b>	Rely upon derived equations from test data; the equations would be expressed in the same mathematical form, ir-regardless of the phase.
Nelson Linear Model	used 104 cost factors of which only 14 were valid; regression results were poor in estimating the generating data base. 4GL applications should look to cost factors that are statistically valid for the data set considered.
Farr and Zagorski Linear models	used regression to develop several equations depending on complexity; different equations may be applicable depending on tool or organization in 4GL. Regression is valid approach to cross-validating and calibrating a predictive model.
Wrigley 4GL size estimation model	generated LOC estimates by reverse engineering 4GL application components; good explanation of variance for both module programs and entire systems; did not estimate LOE.
Halstead Complexity model	estimates complexity of modules based on operators and operands in a module and programmer work independence; 4GL components are not likely to exhibit extensive operational complexity.
Tausworthe Interactive staffing effort model	uses 68 factors and Rayleigh curve to adjust productivity rates; significant expert judgment required; no clear connection to 4GL application structures evident.
Bailey-Basili estimation model	good predictive accuracy in specific aerospace environment of NASA projects; multi-linear regression analysis used to generate a successful estimation model. A targeted model for specific software development language or environments works best to predict costs.
<b>Dynamic Models</b>	Vary with the behavior of the software life cycle; consider staffing changes over life cycle, learning curves, and other factors of change in the life cycle; many or all of the variables are inter-dependent
Norden (IBM) Staffing model	used Rayleigh staffing buildup and declination parameters in engineering applications; Putnam used theory to create model.
Putnam theoretical model and SLIM	looks at staffing, complexity, and schedule; tends to over-estimate medium and large projects; overall accuracy very poor. SLIM also estimates LOC and uses expert judgment in deriving estimates. Business application, the primary use of 4GLs, results in poor estimates, despite calibration.
<b>Hybrid Models</b>	Combine characteristics of expert, static, and/or dynamic models
Basic COCOMO	uses database of 2GL and 3GLs; estimates effort by activities and 15 cost factors; Beta-weights and exponent generated by Boehm's judgment; primary input is estimate of SLOC.
Intermediate and Detailed COCOMO	allows adjustment of Beta-weight and exponent dependent on type of application: organic, semi-detached, or embedded; Detailed adds two cost factors; primary input is still an estimate of SLOC; accuracy for estimating effort is poor in 4GL applications.

Ada-COCOMO	uses estimate of SLOC; effort is calculated based on experience and project management techniques as well as requirements volatility; object-oriented approach to estimation guides development of a more structural model to improve accuracy in 4GL environments.
Pfleeger COSTTOOL Object-Oriented model	used objects and methods to estimate LOE; improved estimation accuracy as compared to conventional LOC models; structural foundation.
Rubin ESTIMACS	uses team questionnaires to develop estimates of effort; does not rely on LOC estimates; wide ranges of error and overall poor explanation of variance in tests.
RCA PRICE-S	uses project factors and complexity to generate costs for phases of development; used to estimate aerospace projects; most 4GLs are used for business application, so this model is likely to not be applicable..
Verner and Tate 4GL estimation model	developed explanation equations, then tested estimates on subsequent increments of projects; large number of projects tested indicates good validity using Conte criteria; high level structural constructs. Did not address cost expressed in LOE.

COSTMO-4GL is intended to be a direct measurement modeling approach, because it could generate a LOE estimate using counts of the instances of the actual application components used in developing software applications in modern software development environments: forms, reports, tables, and modules. While it is noted that careful and systematic measurement of attributes about an application can lead to accurate predictive models, it seems far easier to capture direct measure of what components are being constructed in the application. Such a model would be more intuitively useful if the levels of estimation accuracy are equally acceptable. Industry needs a cost model that can be used to not only explain, but most important to predict the labor costs of software development projects. This model should be built of observable, empirically derived structural components that directly influence LOE.

For the modern software development cost model case study, data on several projects were drawn from official archived documentation, effort estimates and actual post-hoc effort data for 22 management information systems applications constructed for the Technical Management Information System (TMIS) for the Space Station Freedom Program (SSFP) in the 1990s. The basic model of four structural component parameters, COSTMO-4GL, resulted in a mean magnitude of relative error (MMRE) in prediction accuracy of 50%, and the probability of an estimate having 25% or less magnitude of error is 41%. In the best 17 of the 22 cases, the MMRE(.77) was 25% in validation trials.

## **Evolving Recognition of Appropriate Structural Cost Estimation Models**

Finally, with the migration to object-oriented tools and software development environment, the argument for structural cost estimation models becomes even more compelling. It was not until the early 1990s that researchers began to recognize that 4GL software development might be structurally different and require different sizing and costing algorithms [WRIGLEY91] [VERNER92] [TATE92] [EJIOGU91].

Since the structural behavior of the components is essentially consistent across 4GL applications, independent variables selected from those components that are actually what the effort is expended on: forms, reports, tables and modules - seem reasonable from both the theoretical and operational perspective [EJIOGU91]. Pfleeger (PFLEEGER89) developed a model based on objects and methods that provided improved predictive accuracy in object-oriented software development environments. The relatively favorable results of Pfleeger's structural model when compared to expert estimates of lines of code and complexity adjustment factors held out the suggestion that models based on the structural components of a language environment may be more accurate in predicting level of effort than models using lines of code estimates. If the structural component parameters of a 4GL cost model are based upon the same 4GL structural components that make up the application, there is a direct, logical relationship of the cost factors to the application components [EJIOGU91]. This research provides impetus to development and consideration of new structural models for other categories of modern software development environments such as 4GL.

## **Timing of Cost Estimates in the Development Lifecycle**

Furthermore, as we analyze the practice of producing software cost estimates, we must consider at what stage in the overall lifecycle of the application can useful estimates be prepared and still be applied to plan and staff the completion schedules for developing and delivering management information systems. Estimates of cost and effort may have greater accuracy at the conceptual design phase [BOEHM81] [LEDERER92].

At the software definition phase, customer needs are identified and translated to software specifications. Early in the software development lifecycle, the developer is generally not able to describe the structural components of the application that will be built to satisfy user requirements. However, it is also recognized that early on in the software development lifecycle, good estimates of effort are generally not possible. The following Figure illustrates that the error of estimates decrease as the software development project progresses in its development lifecycle.

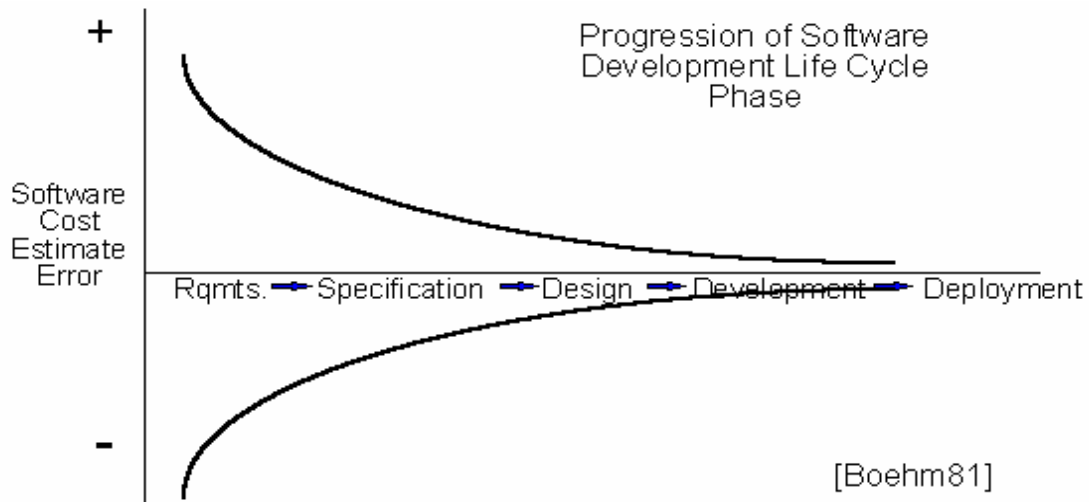


FIGURE 1: Software Effort Estimation Accuracy in the Software Development Life-cycle

During the life cycle requirements analysis phase, analysts work with users and programmers to develop design specifications in terms of the application's forms, reports, tables, and modules. Programmers then use the design specifications to directly identify what needs to be constructed to meet user requirements [LEVIN97]. Naturally, after development has been completed, we have actual cost data and an estimate of expected level of effort to be expended is no longer relevant. So, for estimates to be useful to us, they must be sufficiently accurate to staff projects, yet early enough in the software development lifecycle to make those staffing and resource decisions. Research indicates that estimates of costs and effort can be made with greater accuracy at the conceptual design phase than at any of the planning or requirements definition phases of software acquisition [BOEHM81] [LEDERER92].

### Industry Measure for Cost Model's Predictive Accuracy

Several benchmarks for accuracy are widely used in the software development industry for evaluating the performance of cost estimation models. One of these looks at a cost model's ability to predict discrete applications' level of effort at some level of accuracy  $L$ , where  $k$  out of  $n$  projects are within this range of acceptable accuracy in predictions. This is expressed as

$$\text{PRED}(L) = k/n$$

where  $L$  is the level of predictive accuracy, in terms of the magnitude of relative error

$k$  is some subset of the  $n$  projects falling within that level of predictive accuracy

$n$  is the total number of projects

From this, Conte [CONTE86] conducts empirical studies that suggest  $L = .25$  as the magnitude of relative error and  $k/n \geq .75$  where LOE predictions in at least 75% of the cases fall within 25% of actual LOE as an “acceptable” level of error.

For example, applied in a large software development program of 100 application projects where 25 estimates were off by 200%, 15 were estimated perfectly with 0% error, and 60 projects were estimated with 30% error; only 15 of the 100 projects would be considered to have been acceptably estimated when  $PRED(.25) = .15$  and the Conte criterion would not be met. On the other hand, the predictive accuracy is 30% in the best 75% of the cases  $PRED(.30) = .75$  and the probability of a project being estimated with .30 magnitude of relative error (MRE) is .75.

### **The COSTMO-4GL Cost Estimation Model Study**

The a priori proposition in this research case study was that a weighted linear sum of the structural components forms, reports, tables, and modules could be used to accurately describe or forecast LOE. The coefficients of these structural component parameters should be positive. Thus, if any of the four structural variables increased in number, there would be a related increase in the LOE for the software development application project. From a theoretical perspective, the stepwise linear regression statistical technique was chosen to determine the best estimates of the parameters within the structural model that most accurately explains the relationship between the values of the structural components and LOE. The dependent variable is expressed as the level of effort in person calendar days. The initial model takes the form:

$$LOE_p = a + b_1 * forms + b_2 * reports + b_3 * tables + b_4 * modules$$

Where,  $LOE_p$  is the number of person days of effort it takes to develop a software application project  $p$ ;

$a$  is the  $y$  intercept

$b_1$  is the number of person days it takes to develop one Form  $F$

$b_2$  is the number of person days it takes to develop one Report  $R$

$b_3$  is the number of person days it takes to develop one Table  $T$

$b_4$  is the number of person days it takes to develop one Module  $M$

A non-zero parameter was deemed acceptable for descriptive purposes, and provides a descriptive estimate of effort not represented by the assumed structural model and four variable structural component parameters. Its use in regression analysis is normally regarded as part of the unaccounted for independent variables and may be regarded as “overhead” in a prescriptive model. This overhead could be accounted for in holidays, vacations, sick leave, and other non-development time, such as meetings, reviews, etc. The count of instances of each independent variable (forms, reports, tables, modules), were multiplied by an empirically derived coefficient multiplier  $b_1, b_2, b_3, b_4$  estimated through linear regression analysis, and added to some constant  $a$  for the  $y$  intercept, to

yield a total estimated LOE for project  $p$ . Finally, to test this descriptive model, the resulting predictions of the dependent variable,  $LOE$ , were compared to the actual  $LOE$  for the case study database of 22 software applications.

In the model, COSTMO-4GL, modules are defined to include both modules written in lines of code and modules drawn from a 4GL runtime library. For reasons of simplicity, as well as the assumption that 4GL applications will generally be constructed with a combination of these types of modules, the research cost model uses the assumption of average effort to construct or utilize all modules. This approach is consistent with other researchers in structural models for component-based software development environments [PFLEEGER89] [VERNER91]. The operational definition of the independent variables is given below:

1. Forms are objects created by a developer for interaction or navigation by the user. The 4GL tool provides templates for the developer to use. The form templates can be used as Menus that move the user through the application, or for use as data entry, querying Screens, or other Forms required for the application.
2. Reports are objects that a developer uses to retrieve data from tables, to format and present that data to the user. The 4GL tool provides an easy to use interface and ready-made products for the developer to tailor and create specialized reports.
3. Tables are objects created to store data. The 4GL tool provides features for the developer to define and generate data definition language (DDL) and create tables automatically from design models.
4. Modules represent that portion of a software development application that cannot otherwise be delivered, except to be created. These might be computational algorithms, transaction handling, or processes. The modules may be constructed using third generation languages, such as C+.

A large management information system was used as the basis for obtained actual data on four independent variables for twenty-two 4GL projects. Simple enumeration of the actual instances of each of the four variables was used. The actual number of people dedicated to each distinct application development team was also determined from official NASA records. In general, designs were considered baseline, after the requirements had been accepted by the customer. The research focused on the costs of development beginning with the approval of conceptual design specifications. The actual LOE was calculated by multiplying the full-time equivalent (FTE) staff assigned to the project by a count of days. The count of days was uniformly defined to be that total number of calendar days from the Design Approval Date to the Date of Deployment. For the 22 cases studied, the total LOE ranged from 104 days to 1,107 days.

As a comparison model, expert estimates of effort that had been created by the development organization were used for the same applications. Data was not available to describe specifically the basis for the expert-generated estimates of lines of code, nor for the experts' basis of historical estimation of productivity. The productivity factor (lines of code per programmer month) was based on software development projects that were completed earlier than the 22 projects included in this case study, and compared similarly

to industry standards [JONES91A] for the number of lines of code that can be written each month by a proficient programmer.

## Results of the Research Case Study of 22 Projects

Model accuracy and validity are important considerations in model development. For COSTMO-4GL, two aspects of the proposed model were explored: 1) the multipliers associated with structural component parameters that should be used to describe and predict the cost of developing a 4GL software application [KITCHENHAM92], and 2) the overall predictive accuracy of the model as compared to expert based approaches [LEDERER92]. These were the two goals cited for this research. Successful model validity tests assure future researchers that sufficient statistical analysis has been conducted to demonstrate accuracy of a model for descriptive and prescriptive purposes.

Internal validity is important in demonstrating that the cost factors of the predictive model should demonstrate a logical or reasonable connection to the items being described by the same data set [EJIOGU91]. Since COSTMO-4GL is constructed of the very components in a 4GL software application, there is a direct and reasonable connection in argument. This is known as “construct or content validity” [ADELMAN92]. However, regardless of how valid a model may appear to be in reasonableness, it must also be validated by collecting evidence and analysis of that evidence to demonstrate that the model is correct [PFLEEGER89].

The LOE results of applying the predictive equation are compared to the actual recorded LOE to develop each of the application projects. A relative error is calculated by the equation:

$$RE_p = \frac{(A_p - E_p)}{A_p}$$

where,  $A_p$  is the actual level of effort in person days, for the project  $p$   
 $E_p$  is the estimated level of effort in person days, calculated by multiplying the actual count of cost factors, by the multipliers, and adding the constant value, expressed in person days

If the actual and estimated LOE are equal, there is no relative error (RE). It is worthwhile to note that an incorrect assumption about the predictive accuracy of a model may be drawn when the sum of all RE values for a group of projects is evaluated. In the case where large over-estimates and large under-estimates are present in a group of projects, the values cancel each other out, providing a zero average error for cases where individual error magnitudes are large. To remove this deficiency of average error as a measure, the *absolute value* of the relative errors is used. This is expressed as:

$$|MRE| = \frac{|(A_p - E_p)|}{A_p}$$

where,  $|(A_p - E_p)|$  is the absolute value of the relative error in the actual level of effort and the estimated level of effort in person days

When these absolute values are summed and an average of relative error is calculated, the following equation is used:

$$MMRE = \frac{\sum |MRE|}{n}$$

where, MMRE is the average *amount* or magnitude of relative error across a group of  $n$  projects. The mean is expressed as the calculated average amount of error in the estimated level of effort as opposed to the actual level of effort in person days

This calculated statistic is then used as an evaluation criterion to estimate the error resulting from the use of an estimator model. Taken alone the MMRE should overall be no greater than 25% when averaging over all the cases in the study [VERNER92] [CONTE86]. However, as indicated earlier in this paper, taking this measure and further analyzing the results of relative error across the best 75% of the projects in a group of projects, we will also use the equation PRED(.25) as a measure of “goodness” of the model.

Another measure of goodness was also applied, where the best 75% of the 22 cases are evaluated to determine if these “best predictions” fall within 25% accuracy for effort estimation. It is recommended that this measure be used as a means for determining model calibration or adjustment procedures that were applied in this research. So this measure  $MMRE(.75) = L$  was used in combination with MMRE (for all cases overall), PRED(L) where  $L = .25$  or some other level determined based on level of accuracy requirements for an organization. In this case study, with an applied COSTMO4GL model against the best 75% of the 22 cases, the MMRE was 28%.

### **Regression Analysis and Cross-Validation Studies of COSTMO-4GL Model**

The amount of variance explained by the proposed independent variables is given by the multiple- $r$  squared. A model should also be considered in light of the statistical significance of the proposed independent variables by the  $t$ -tests and the overall statistical significance of the resulting regression equation with the  $F$ -test. A correlation matrix was calculated to capture the inter-correlations between the dependent variable, *LOE*, and the independent variables, *forms*, *reports*, *tables*, and *modules*, as well as inter-correlations between the independent variables.

TABLE 2: Correlation Matrix

	<i>LOE</i>	<i>FORMS</i>	<i>REPORTS</i>	<i>TABLES</i>	<i>MODULES</i>
<i>LOE</i>	1.0000				
<i>FORMS</i>	0.3798	1.0000			
<i>REPORTS</i>	0.4880	0.4755	1.0000		
<i>TABLES</i>	0.2391	0.1915	0.4095	1.0000	
<i>MODULES</i>	0.4654	0.2979	0.1437	0.2097	1.0000

The range of the Pearson Product Moment correlations between the dependent variable and the four proposed independent variables is .2391 to .4880. Two of these correlations are statistically significant at the .05 alpha levels with .4880 for *Reports:LOE* and .4654 for *Modules:LOE*. Inter-correlation between two of the independent variables at .4755 for *Reports:Forms* is statistically significant at the .05 alpha level. This may indicate that there is a correlation between these development activities. This would, in fact, be logical insofar that in producing forms for input, reports often mirror or reflect the users' view of output of the data entered into the specific forms in the software application.

A regression analysis was performed using the four variable model resulting in the initial research model equation:

$$LOE_p = 238.64 + .36 * F_p + 15.06 * R_p - .42 * T_p + 3.19 * M_p$$

where, *F* is the # of Forms in project *p*  
*R* is the # of Reports in project *p*  
*T* is the # of Tables in project *p*  
*M* is the # of Modules in project *p*

The relatively large value of the constant parameter, about half the size of the average LOE (which ranged from 104 to 1,107 days) across all 22 cases, defeats this model initially for use as a prescriptive model [SAGE97]. Also, the equation for LOE estimate indicates that there is a negative relationship between Tables and LOE. It implies that adding more tables would reduce the cost of the project.

### Additional COSTMO-4GL Variant Studies

To isolate the possibility that one or more of the proposed independent variables may provide acceptable levels of predictive accuracy for this group of projects, a backward stepwise linear regression technique was performed with actual, post-hoc enumeration of forms, reports, tables, and modules for each application [ROCHE94]. The backward stepwise regression technique resulted in only two independent variables remaining in the cost estimation regression equation. This equation, generated from the original 22 application projects is stated as:

$$LOE_p = 245.60 + 15.97 * R_p + 3.30 * M_p$$

The backward stepwise technique revealed that the amount of linear variation accounted for by two independent variables was virtually the same as that using the four variables. The multiple-*r* for the equation generated with only the Reports and Modules as independent variables was .6305, compared to the four-variable model with .6342. This is the most parsimonious model that explains the maximum amount of variance in the dependent variable. This equation, generated from the original 22 application projects is stated as:

$$LOE_p = 245.60 + 15.97 * R_p + 3.30 * M_p$$

It was then tested whether this more parsimonious model could yield acceptable levels of predictive accuracy in cross-validation trials [CONTE86], by using the MMRE in 100% of the cases, as well as the best 75% of cases, and the PRED(.25) level, as described earlier in this paper. Analysis of our data set of 22 observations was somewhat ambiguous. A larger data set would clearly be desirable in further work and is actively being sought. The following table displays the actual LOE in person days as well as the value of predicted LOE for each project, generated by the two-variable cost estimation model. The residuals displayed are the absolute value differences between the actual person days and the estimated person days for each project, based on the application of the COSTMO-4GL model.

Table 3: Actual and Predicted Level of Effort for 22 Projects

<i>Observation</i>	<i>Actual LOE</i>	<i>Predicted LOE</i>	<i>Absolute Value of Residuals</i>
1	104	303.9333	199.9300
2	143	442.1018	299.1010
3	162	358.4330	196.4330
4	200	300.0950	100.0950
5	221	417.3087	196.8087
6	264	368.8728	104.8728
7	324	371.0984	47.0984
8	354	541.7365	188.2366

9	435	345.7677	89.2323
10	450	500.9774	50.9774
11	472	655.6495	183.6495
12	480	1013.8169	533.8169
13	654	525.6953	128.3046
14	712	509.1917	202.8083
15	808	607.8264	200.1736
16	815	793.2803	21.7197
17	830	912.7195	82.7196
18	872	749.2956	122.7044
19	896	686.4313	209.5687
20	919	493.3007	425.1993
21	1001	564.3041	437.0044
22	1107	759.9725	347.0275
<hr/>			
Sums	12222	12221.8085	4367.49

In evaluating the predictive accuracy of the model to a set of actual counts of components for the 22 cases that generated the COSTMO-4GL model, Conte's level of effort predictive accuracy criterion PRED was applied. The application of this criterion in the two-variable model resulted in only 9 of the 22 projects having a magnitude of relative error less than 25%. This would not meet the Conte criteria expressed as  $PRED(.25) \geq 75\%$ . This two variable model applied to the 22 cases resulted in  $PRED(.25) = 41\%$  and a MMRE of 54%.

To conduct an analysis of variance on the behavior of the model, cross validation was performed on twelve different subsets of the 22 case data set. It is clear that a model that cannot reasonably describe the dependent variable in the generating data asset is of little or no use in predicting the dependent data a priori [NELSON66]. The data sets were split into various groups from the generating data set. Trials were performed to create predictive COSTMO-4GL models for subsets of odd and even-numbered groups of 11 cases used to predict the other 11 cases, and to predict other randomly selected groups of 6 cases. This entailed using new regression-generated equations, developed from the odd and even subsets, to estimate the LOE of groups of cases. This synthetic predictive cross validation was planned in several sample groupings to simulate cross validation checks for consistency in estimation accuracy levels, and for model stability and robustness. The following table presents the results of these cross-validation trials.

Table 4: Predictive Accuracy in Cross-Validation Trials using the Two-Variable COSTMO-4GL Model

<b>FULL 22 CASES</b>	
MMRE in 100% of 22 cases	predictive accuracy within 54%
MMRE in 75% of 22 cases	predictive accuracy within 28%
PRED(.25)	41% of 22 cases
<b>11 ODD-NUMBERED CASES</b>	
MMRE in 100% of 11 cases	predictive accuracy within 53%
MMRE in 75% of 11 cases	predictive accuracy within 41%
PRED(.25)	27% of 11 cases
<b>11 EVEN-NUMBERED CASES</b>	
MMRE in 100% of 11 cases	predictive accuracy within 62%
MMRE in 75% of 11 cases	predictive accuracy within 24%
PRED(.25)	36% of 11 cases
<b>6 CASE - Random Trial 1</b>	
MMRE in 100% of 6 cases	predictive accuracy within 63%
MMRE in 75% of 11 cases	predictive accuracy within 28%
PRED(.25)	17% of 6 cases
<b>6 CASE - Random Trial 2</b>	
MMRE in 100% of 6 cases	predictive accuracy within 47%
MMRE in 75% of 11 cases	predictive accuracy within 22%
PRED(.25)	17% of 6 cases
<b>6 CASE - Random Trial 3</b>	
MMRE in 100% of 6 cases	predictive accuracy within 73%
MMRE in 75% of 11 cases	predictive accuracy within 25%
PRED(.25)	33% of 6 cases
<b>6 CASE - Random Trial 4</b>	
MMRE in 100% of 6 cases	predictive accuracy within 47%
MMRE in 75% of 11 cases	predictive accuracy within 20%
PRED(.25)	33% of 6 cases
<b>6 CASE - Random Trial 5</b>	
MMRE in 100% of 6 cases	predictive accuracy within 73%
MMRE in 75% of 11 cases	predictive accuracy within 28%
PRED(.25)	33% of 6 cases
<b>11 CASE - Random Trial A</b>	
MMRE in 100% of 11 cases	predictive accuracy within 51%
MMRE in 75% of 11 cases	predictive accuracy within 26%
PRED(.25)	27% of 11 cases
<b>11 CASE - Random Trial B</b>	
MMRE in 100% of 11 cases	predictive accuracy within 49%
MMRE in 75% of 11 cases	predictive accuracy within 27%
PRED(.25)	36% of 11 cases
<b>11 CASE - Random Trial C</b>	

MMRE in 100% of 11 cases	predictive accuracy within 57%
MMRE in 75% of 11 cases	predictive accuracy within 19%
PRED(.25)	55% of 11 cases
<b>11 CASE - Random Trial D</b>	predictive accuracy within 47%
MMRE in 100% of 11 cases	predictive accuracy within 28%
MMRE in 75% of 11 cases	27% of 11 cases
PRED(.25)	
<b>11 CASE - Random Trial E</b>	predictive accuracy within 36%
MMRE in 100% of 11 cases	predictive accuracy within 28%
MMRE in 75% of 11 cases	27% of 11 cases
PRED(.25)	

### Comparator Expert-Based Estimation

One of the goals of this research was to determine if a structural model produces better accuracy in estimating software effort than other models in use. Since expert estimation may be the most broadly used estimation procedure in the industry [LEDERER92] [LEVIN97], it is appropriate to use data collected on expert estimates related to the same 22 application projects in the same program to compare levels of accuracy.

During the course of the NASA program, experts were asked to produce estimated sizes of the individual applications, expressed in Estimated Source Lines of Code (ESLOC), to garner an overall size for the management information system being developed. As part of the expert estimation exercise, productivity measures had been developed and are used in this comparative study to give equivalent expert-based LOE estimates. The results of the expert-based estimates of source lines of code (ESLOC) estimation model were analyzed and compared to COSTMO-4GL for accuracy in estimating LOE. The correlation, shown in the table below, between the Actual LOE and the Estimated LOE based on the expert estimate model used on these projects is .2477 with an  $r$ -squared ( $r^2$ ) of .06.

TABLE 5: Results of Regression of Expert-Based Estimation Model on 22 Cases

<b>F - TESTS OF STATISTICAL SIGNIFICANCE</b>					
	<u>df</u>	<u>SS</u>	<u>MS</u>	<u>F</u>	<u>Significance F</u>
Regression	1	127411.8	127411.8	1.3073	0.2664
Residual	20	1949156	97457.8		
Total	21	2076568			

## ANALYSIS OF VARIANCE

	<u>Coefficients</u>	<u>Standard Error</u>	<u>t Stat</u>	<u>P-value</u>
Intercept	453.0550	111.6392	4.0582	0.0006
SLOC/pm	0.0255	0.0223	1.1434	0.2664

The related  $F$ -test clearly indicates there is no significant linear trend in the expert-based estimation model. With one degree of freedom for 22 cases,  $F$  equals 1.3073, with an alpha level of .2664. The  $t$ -statistics for the Beta-weight from the regression are shown in the table below, and do not indicate statistical significance. The much larger intercept at 453, as compared to COSTMO-4GL, also is a further indication that this model does not sufficiently explain variance in the LOE. This is consistent with other source lines of code (SLOC) models and expert-based judgments of size expressed in lines of code (LOC) and translated to LOE.

The estimated LOE for each application project included in the COSTMO-4GL study is shown below and compared to the actual LOE captured for each of the projects. The value of the residuals (or M.R.E.) for each of the projects shown is in Table 6. The magnitude of the relative errors clearly indicates that the expert estimation model used for these projects tends to significantly over-estimate the LOE when compared to the actual LOE reported for the application projects. As well, the overall LOE for the projects was overestimated by seven times the actual total LOE or 19,284%, rendering this cost estimation approach highly inaccurate. The expert based ESLOC estimation model does not meet the Conte criterion for effort estimation accuracy.

TABLE 6: Experts' Estimates LOE Compared to Actual LOE for 22 Projects

Observations	Actual Dependent LOE	Expert Estimated Dependent LOE	MRE  %
1	104	2617	2416%
2	143	1856	1198%
3	162	2617	1516%
4	200	3846	1823%
5	221	2444	1008%
6	264	2617	891%
7	324	2617	708%
8	354	8292	2246%
9	435	3846	784%
10	450	5076	1028%
11	472	5076	975%
12	480	4482	834%
13	654	1856	184%
14	712	1656	133%
15	808	3271	305%
16	815	8292	917%
17	830	1656	99%
18	872	3271	275%
19	896	3094	245%
20	919	2444	166%
21	1001	14979	1396%
22	1107	2622	137%
TOTALS	12,222	88,526	19,284%
MMRE for 100% of cases =			877%
PRED for best 73% of cases =			582%
PRED(.25) =			0%

In summary, the mean magnitudes of relative error render the expert estimation approach largely unworkable for effort estimation, even in descriptive situations. The expert-based model does not exhibit a linear trend, and it is highly questionable whether the expert-based model, using ESLOC as a means of producing estimates, can be calibrated to improve accuracy.

## Conclusions and Recommendations

The goal for developing a new model based on structural components has been met for this group of 4GL applications, though not able to meet the goal of achieving acceptable level of accuracy in the 22 cases studied. The linear model tested on the 22 cases proved robust and stable through cross validation and verification trials on software development applications. Additional studies were executed with the four variable model forced to zero with the result that predictive accuracy was not improved using the simple counts of four structural component parameters. When applied to the 22 cases, the COSTMO-4GL two variable model gave similar levels of accuracy/error as demonstrated in 12 cross-validation trials. Another measure of goodness for cost estimation models was discussed and presented. In the best 75% of the 22 cases, the MMRE was 28%.

COSTMO-4GL is a major departure from the estimating techniques used historically to determine the a priori cost of software development in modern software development environments. However, the results of the analysis indicate that expressing LOE for these 22 4GL software development projects by simply enumerating forms, reports, tables and modules is not sufficient to give acceptable cost estimates, according to industry standards. This may, in part, be due to the fact that the data set size was not sufficiently large to estimate parameters in the four variable and two variable COSTMO-4GL models adequately.

This approach results in a cost estimate that is expressed in terms of the LOE in person days, rather than LOC, function points, delivered source instructions (DSI), or some other artifact metric. The advantage of this approach would be that an organization's dollar cost could be directly estimated from person days. There would be no need to translate LOC or function points based on an assigned or derived productivity factor into LOE. The COSTMO-4GL model is a procedural approach to build prescriptive cost models, not a set of coefficients that can be migrated and applied directly in all cases. However, the actual model of four structural parameters is descriptive. Calibration is likely to be essential since the levels of accuracy were not within acceptable accuracy range.

Further research with larger data sets is required to stabilize and isolate more closely the combination of cost factors, in addition to the four structural components identified in COSTMO-4GL that contribute to modern software development using 4GL software tools and their associated structural components. Careful attention must be paid to the choice of a model based upon the structural composition of the applications, the specific modern software development tool used, and the combination of components estimated from conceptual design specifications. Furthermore, this study demonstrates that application of a linear model utilizing backward stepwise regression provides an acceptable approach to testing and calibrating a structural model appropriate to the organization that produces modern software applications. Finally, this research did achieve the goal to demonstrate that the structural approach to identifying cost drivers is an appropriate methodology for producing estimates of software development level of effort costs.

## BIBLIOGRAPHICAL REFERENCES

[ADELMAN92] Adelman, L, Tolcott, M.A., and Bresnik, T.A., "Examining the Effect of Information Order on Expert Judgment", *Organizational Behavior and Human Decision Processes*, 1992.

[ALBRECHT83] Albrecht, A.J., and Gaffney, J.E., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", *IEEE Transactions on Software Engineering*, Vol. SE9, No. 6, November 1983, pp. 639-648.

[BAILEY81] Bailey, J.W., and Basili, V.R., "A Meta-Model for Software Development Resource Expenditures", *Proceedings of the 5th International Conference on Software Engineering*, 1981, pp. 107-116.

[BOEHM81] Boehm, B.W., Software Engineering Economics, Prentice-Hall, New Jersey, 1981.

[BRYNJOLFSSON93] Brynjolfsson, E., "The Productivity Paradox of Information Technology", *Communications of the ACM*, Vol. 36, No. 12, December 1993, pp. 66-77.

[BUSWK76] Unknown, "RCA's Uncanny System for Estimating Costs", *Business Week*, Vol., No., June 7, 1976.

[CONTE86] Conte, S.D., Dunsmore, H.E., and Shen, V.Y., Software Engineering Metrics and Models, Benjamin/Cummings Publishing Co., Inc., Menlo Park, CA 1986.

[CUELENAERE87] Cuelenaere, M.J., van Genuchten, I.M., and Heemstra, F.J., "Calibrating a Software Cost Estimation Model: Why and How?", *Information and Software Technology*, Vol. 29, No. 10, December 1987, pp. 558-567.

[CURTIS81] Curtis, B., "The Measurement of Software Quality and Complexity", *Software Metrics: An Analysis and Evaluation*, (A Perlis, et al.), MIT Press, Cambridge, MA 1981, pp. 203-224.

[DEMARCO82] Demarco, T., Controlling Software Projects: Management, Measurement and Estimation, Yourdon Press, 1982.

[EJIOGU91] Ejiogu, L.O., Software Engineering with Formal Metrics, QED Publishing Group, Boston, MA, 1991.

[FENTON94] Fenton, N., "Software Measurement: A Necessary Scientific Basis", *IEEE Transactions on Software Engineering*, Vol. 20, No. 3, March 1994, pp. 199-206.

- [GAFFNEY84] Gaffney, J.E., Jr., "Estimation of Software Code Size based on Quantitative Aspects of Function Points, "Journal of Parametrics", Vol. 4, No. 3, September 1984, pp. 23-33.
- [GAFFNEY96] Gaffney, J., Software Cost Estimation Using Simplified Function Points, presentation to DoD Software Technology Conference, Salt Lake City, Utah, April 25, 1996, pp. 1-15.
- [HALSTEAD77] Halstead, M., Elements of Software Science, Elsevier Press, Amsterdam, 1977.
- [JONES86] Jones, C., Programming Productivity, McGraw-Hill, New York, NY, 1986.
- [JONES91A] Jones, C., Applied Software Measurement, McGraw-Hill, New York, NY, 1991.
- [KEMERER87] Kemerer, C.F., "An Empirical Validation of Software Cost Estimation Models", Communications of the ACM, Vol. 30, No. 5, May 1987, pp. 416-429.
- [KEMERER93] Kemerer, C.F., "Reliability of Function Points Measurement, A Field Study", Communications of the ACM, Vol. 36, No. 2, February 1993, pp. 85-97.
- [KITCHENHAM92] Kitchenham, B., "Empirical Studies of Assumptions That Underlie Software Cost Estimation Models", Information & Software Technology, Vol. 31, No. 1, pp. 211-218.
- [KLEIN95] Klein, D., "Developing Applications with the Alpha UIMS", Interactions, Vol. II, No. 4, October 1995, pp. 48-65.
- [KLEPPER95] Klepper, R., and Bock, D., "Third and Fourth Generation Language Productivity Differences", Communications of the ACM, Vol. 38, No. 9, September 1995, p. 69-79.
- [LEDERER92] Lederer, A.L. and Prasad, J., "Nine Management Guidelines for Better Cost Estimating", IEEE Transactions in Software, February 1992, pp. 51-59.
- [LEVIN97] Levin, R., "Programming Goes Prefab", Information Week, January 13, 1997, pp. 36-48.
- [MYERS86] Myers, E., "A Natural Fit? (Data base Machines and Fourth Generation Languages)", Datamation, vol. 32, January 15, 1986, p. 28.
- [NELSON66] Nelson, E.A., Management Handbook for the Estimation of Computer Programming Costs, System Development Corporation, Santa Monica, California, October 31, 1966.

[PALMER90] Palmer, J.D., and Pfleeger, S.L., "Software Estimation for Object-Oriented Systems, September 26, 1990, originally presented at IFPUG, presented at George Mason University, Fairfax, Virginia, Spring 1991.

[PFLEEGER87] Pfleeger, S.L., Software Engineering: The Production of Quality Software, MacMillan, New York, NY 1987.

[PFLEEGER89] Pfleeger, S.L., "An Object-Oriented Approach to Software Cost Development", (Ph.D. dissertation), School of Information Technology, George Mason University, Fairfax, VA, 1989.

[PFLEEGER93] Pfleeger, S.L., "Lessons Learned in Building a Corporate Metrics Program", IEEE Software, Vol. 10, No. 3, May 1993, pp. 67-74.

[PFLEEGER95B] Pfleeger, S.L., "Experimental Design and Analysis in Software Engineering, Part 5: Analyzing the Data", ACM Software Engineering Notes, Vol. 20, No. 5, December 1995, pp.14-17.

[PUTNAM84] Putnam, L.H., Putnam, D.T., and Thayer, L.P., "A Tool for Planning Software Projects", Journal of Systems and Software, Vol. 5, January 1984. pp. 147-154.

[PUTNAM80] Putnam, L.H., Tutorial Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers, IEEE Computer Society Press, New York, New York, 1980.

[ROCHE94] Roche, J.M., "Software Metrics and Measurement Principles", ACM Software Engineering Notes, Vol. 19, No. 1, January 1994, pp. 77-85.

[SAGE97] Sage, A.P., series of meetings at George Mason University, School of Information Technology and Engineering, April 1997, Fairfax, Virginia.

[TATE91] Tate, G., and Verner, J., "Approaches to Measuring Size of Application Products with CASE Tools", Information and Software Technology, 1991, pp.622-628.

[TATE92] Tate, G., Verner, J., and Jeffery, R., "CASE: A Testbed for Modeling, Measurement and Management", Communications of the ACM, Vol. 35, No. 4, April 1992, pp. 65-72.

[TINNIRELLO91] Tinnirello, P.C., "OOPS, I'm in a 4GL, So Get Off My CASE", Information Week, October 21, 1991, p. 92.

[VAN GENUCHTEN91] Van Genuchten, M., and Koolen, H.J.A., "On the Use of Software Cost Models", Information & Management, Vol. 21, No. 1, August 1991, pp. 37-44.

[VERNER91] Verner, J., and Tate, G., "Approaches to Measuring Size of Application Products with CASE Tools", Information and Software Technology, Vol. 33, No. 9, November 1991, pp. 622-628.

[VERNER92] Verner, J., and Tate, G., "A Software Size Model", IEEE Transactions on Software Engineering, Vol. 18, No. 4, April 1992, pp. 265-278.

[WALSTON77] Walston, C.E., and Felix, C.P., "A Method of Programming Measurement and Estimation", IBM Systems Journal, Vol. 16, No. 1, 1977, pp. 54-73.

[WRIGLEY91] Wrigley, C.D., and Dexter, A.S., "A Model for Measuring Information System Size", MIS Quarterly, Vol. 15, No. 2, June 1991, pp. 245-257.