Comput. & Indus. Engng Vol. 9, No. 2, pp. 165–174, 1985
Printed in the U.S.A.

0360-8352/85 $3.00 + .00
© 1985 Pergamon Press Ltd.

A SIMULATION APPROACH FOR THE ANALYSIS AND FORECAST OF SOFTWARE PRODUCTIVITY

JORGE L. ROMEU
Data and Analysis Center for Software, IIT Research Institute, Rome, NY 13440, U.S.A.

(*Received* 4 *April* 1984; *received for publication* 20 *August* 1984)

Abstract—A simulation model approach for the forecast of software effort and productivity is presented. The approach represents an industrial engineering solution to a software engineering problem. It is based on a statistical model that describes the software variable size and effort as a product of a signal plus random noise. A roadmap for its construction, validation and operation is developed. Its advantages and disadvantages with respect to other existing procedures are discussed. Finally, its implementation and use are illustrated with a real-life numerical example.

INTRODUCTION

During a study of the effects of programming practices on software development efforts performed at the Data and Analysis Center for Software (DACS),† a nonparametric analysis approach was proposed as a more efficient procedure (in our specific context) than the traditional parametric approaches. The efficiency was assessed on the grounds of the nonparametric procedure's ability to filter out noise in the data.

After a long and thorough study, it was determined that these noisy data were the consequence of some inherent characteristics of the software activity. It was concluded that these characteristics had the effect of lowering the data measurement scale to a level below the one required for the appropriate use of the parametric statistical procedures. A simple statistical model was then derived for description purposes[9]. In addition to a theoretical discussion and an in-depth analysis of software data, a simulation model was conceived and implemented with the objective of providing understanding of, and empirical support for, the analysis in question.

The model was implemented and validated, and a sensitivity study was conducted. It is now being proposed as an approach by which, when extended/calibrated to a specific environment, the model may be used as an alternative analysis tool in the forecasting of effort and productivity.

This paper addresses the problem of the implementation, validation and use of such a simulation approach and discusses its advantages and disadvantages with respect to other existing procedures.

BACKGROUND

Accurate estimation of a software project size, the effort required to produce it and the probable productivity level attained during the production process are three relevant objectives of software engineering. This is especially important at very early stages of software project development. If a good estimate is available at the requirements or early design phase, the software engineer may staff, schedule, budget and plan ahead with great accuracy. Unfortunately, this still does not occur. Initial software cost estimations may range within a factor of 4 in the initial conceptual phase and up to 1.5 in the software requirements specifications phase[1].

The existing forecasting procedures for project effort and productivity include regressing these values on a function of the estimated size. Usually, two regression pro-

† The DACS provides a centralized source for current, readily usable data and information concerning software technology. Among its objectives are to encourage the diffusion of technology, to provide data for research, to improve the transfer of software engineering technology, minimizing duplication efforts, and to provide analysis services to the Department of Defense, industry and academia.

cedures have been used: parametric and nonparametric linear regressions. Our analyses experience at the DACS using these procedures on software engineering data can be summarized as follows:

(i) Parametric regression: Given the measurement level problems detected and the numerous and large violations of the regression model's assumptions found during the analysis of the DACS Productivity Data,† no safe confidence intervals nor point estimation could be obtained with these procedures.

(ii) Nonparametric regression: This procedure improves the previous one because it is designed to deal with very noisy data (i.e. large outliers). Its estimators are more stable and robust to the violations in the assumptions of the usual parametric regression, hence better. Nevertheless, this procedure cannot provide confidence intervals for these estimations.

However, the state-of-the-art in software cost estimation and the amounts involved in software development are such that justify the extensive research being conducted in this area and the numerous existing estimation procedures.

The simulation approach proposed here builds upon the previous methods and, in addition, provides an empirical confidence region for the pair (development size, effort). As a consequence, an implicit empirical confidence interval for project productivity is also obtained. This approach provides an alternative that may be used concurrently with the other mentioned tools to refine the initial estimations or perform sensitivity analyses.

This opportunity to perform sensitivity analyses on the initial estimates constitutes, perhaps, its main feature.

THE SIMULATION MODEL

The measurement problem

Grossly speaking, a measurement scale is the standard to determine a given characteristic. It can attain four increasing levels of strength: nominal, ordinal, interval and ratio.

The level of a measurement scale is nominal when the characteristics are given only in categories, e.g. white or black. It is ordinal when these categories can be ordered by a criterion, e.g. small, medium or large. It is an interval scale when it preserves the distance between two points even though the "zero" of the scale may vary for different scales measuring the same variable, e.g. the Kelvin and Fahrenheit temperature scales. Finally, it is a ratio scale if the zero is an absolute value, e.g. zero mass.

Statistical procedures are defined for different scales and can be correctly applied up to the scale for which the procedure is defined (see Fig. 1). For example, contingency tables are defined for nominal scale variables, rank tests are defined for ordinal scale variables and parametric tests that consider the calculation of distances (means, variances, residuals, etc.) are defined for variables given in at least an interval level of their measurement scale[2, 3]. It is possible to lower the level of the measurement scale (i.e. take gross income data to income bracket data) and lose information in the process. The inverse process is not possible (i.e. take bracket information and specify income) without additional information.

Interpretation and model description

The conceptual basis of the present simulation lies in the measurement problem described above. Being unable to estimate the different factors that cause the level of the measurement scale to go down, we propose to consider the joint effect of these (undesired) factors as a random variable. The (prior) distribution of this variable will be obtained with the help of existing data and past experience of the software engineers. The process can be explained in the following way:

If the programming activities had achieved an advanced stage of development so as

† DACS Productivity Dataset contains summary information from over 400 software projects (productivity and error data, project duration, total effort, language and usage implementation technologies).

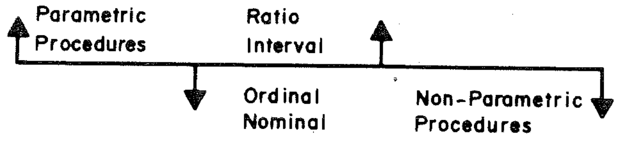


Fig. 1. Measurement scale levels and statistical procedures.

to be sufficiently standardized, any given project would require a standard size x that would exclusively reflect the overall complexity of the problem. This size would therefore be independent of the characteristics of the developing organization, the soliciting organization or the environment. Project sizes (X) would rise following a statistical distribution F (unspecified) in the same manner that heights in human conglomerates follow a (normal) distribution.

At present, owing to the state-of-the-art of the software activity, the size of a project also depends upon, in addition to the intrinsic complexity of the problem, the characteristics of the contracting organization, the developing organization and the environment (Fig. 2).

It is not possible to estimate the combined effect of all these (undesired) factors and remove them from the equation. The present simulation model approach proposes that the overall (random) effect be pooled together into a random noise.

Let this overall "disturbance" or "noise" that modifies the theoretical size of a project be U_x, a random variable dependent on X and distributed G (Table 1). Then, the "observed" or actually achieved size of a project, as modified by all the mentioned factors, is $W = X + U_x$. Notice again that if it were not for the effect of these factors (i.e. application, tools, development team, etc.), the "observed" size W would be equal to the "theoretical" size X, which represents the intrinsic complexity, and the measurement scale problems described before would not exist.

Let Y, the "theoretical" project effort, be, in fact, functionally related to the "the-

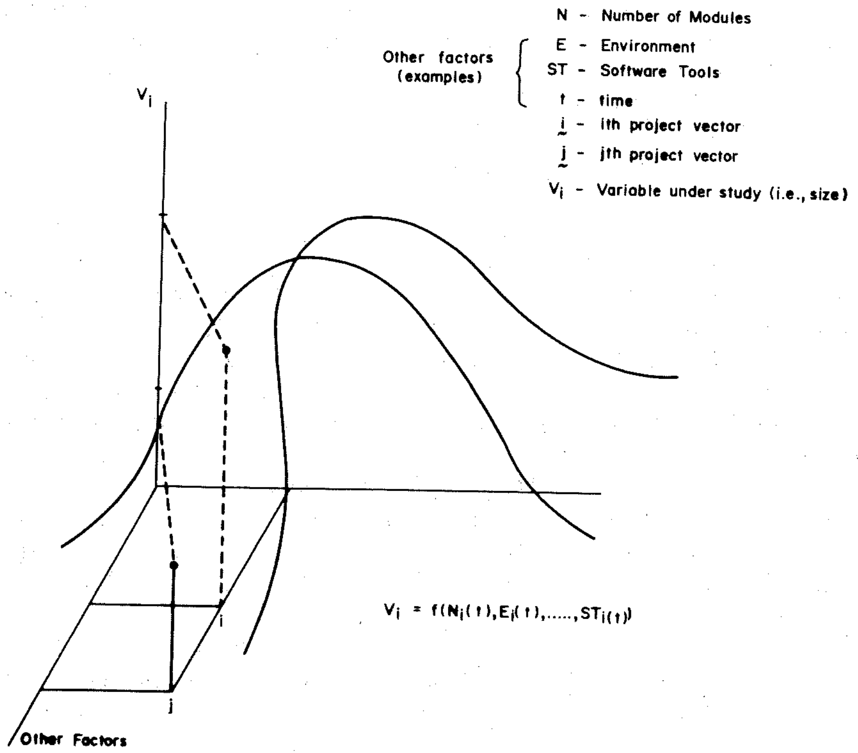


Fig. 2. Representation of the problems occurring when some factors are left out of the analysis.

Table 1. Simulation model

- ''Real'' project sizes X follow a distribution F.
- Measurement scale problem is due to a noise U_x distributed G and dependent on X.
- ''Observed'' sizes W follow: $W = X + U_x$.
- ''Real'' project effort Y is functionally related to ''real'' project size by: $\ln Y = \alpha + \beta \ln X + \epsilon$
- ϵ is a random error distributed H.
- ''Observed'' effort Y' follows: $\ln Y' = \alpha + \beta \ln X + \epsilon_{y'}$.
- $\epsilon_{y'}$ is a random noise distributed L and functionally dependent on Y.

oretical'' project size by the relation

$$\ln Y = \alpha + \beta \ln X + \epsilon,$$

where ϵ is a noise distributed about the ''exact'' value ($\ln Y$) and attributed to human differences in productivity, organizational efficiency, etc. Hence, $\ln Y$ would be the logarithm of the ''real'' effort (if the measurement problem did not exist). Finally, let the ''observed'' (or attained) effort Y' be defined by

$$\ln Y' = \alpha + \beta \ln X + \epsilon_{y'},$$

where $\epsilon_{y'}$, the ''noise'' caused by the aforementioned factors, is similar in spirit and logic and functionally dependent on its related variable Y, as U_x is on X, and distributed L.

In summary, since the ''theoretical'' (unspecified) values (X, Y) of a development project are always unknown, in practice the ''observed'' measurements (W, Y') constitute the data available for analysis in a given environment.

The objective of this simulation approach is that given a value X of (initial) estimated project size, the simulator generates enough pairs (W, Y') with the aid of the distributions describing the noises U_x and $\epsilon_{y'}$ to generate the set (W, Y'). Then, through the plotting and analysis of this set, the analyst can determine an empirical confidence region for final project size and effort (and a confidence interval for the productivity) of the given project. Finally, from the comparison of several alternative sets (W, Y') generated via different distributions of U_x and $\epsilon_{y'}$, sensitivity analyses may be performed and different software engineering criteria may be compared.

Model construction

The extension/calibration of the simulation model to a given environment is centered about three activities: (i) the selection of specific forms for the statistical distributions F, G and L and the estimation of their corresponding parameters; (ii) the selection of a specific functional form for the trend of $\ln Y$ on $\ln X$ and the estimation of its parameters; (iii) the validation of the model under construction. Each of these activities will be developed in the following sections to illustrate a practical implementation of this approach.

Selection of the distribution and trend. To implement the simulation approach, the COBOL subset of the DACS Productivity Dataset was chosen.† A simulation program was written in FORTRAN, using the IMSL routines for generating all statistical distributions, and implemented in a Honeywell DPS8-44D mainframe using the GCOS-3 operating system.

In the process of selecting the distribution F, study of the histograms of the raw and log transformation of the original data and estimation of its means, medians and higher moments were helpful. The goodness of fit (GoF) tests of the original data to a selected

† DACS COBOL subset comprises 28 projects containing information about final project size and total development effort.

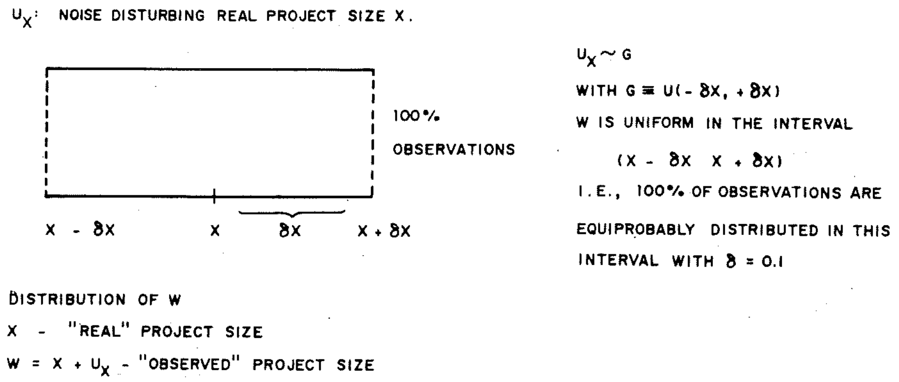
set of statistical distributions also helped to determine the final choice. In our case, log-normal, gamma and exponential distributions were strongly considered. The last was finally selected based upon the simplicity of its parameters and its good fit to the raw data. The mean of this exponential distribution was estimated from the COBOL data, and the specification of F was completed.

Noise U_x was specified by studying the variation in sizes for projects having attained similar efforts. This procedure was repeated at various effort levels and a function fitted through. It was then assumed that an equiprobable distribution, with a spread of 10% about the initial estimated size X, would be a reasonable statistical description (or educated guess) for the distribution G—hence, the selection of the uniform distribution in $(-0.1X, 0.1X)$ as G (see Fig. 3A).

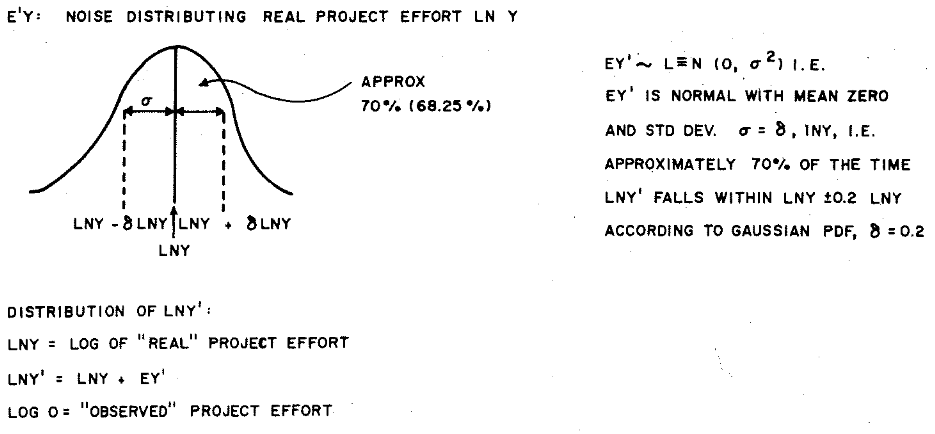
The distribution L of the noise disturbing the "real" effort Y was specified in a similar manner to U_x. In addition to past experience and data analysis, a concurrent fit of the variability in effort, given size, for different levels of size may be performed. In our study, a normal distribution with standard deviation of 20% about the "real" effort was selected for ϵ (see Fig. 3B).

For the relation ln (effort) on ln (size), a linear trend was selected. Using the non-parametric regression approach of Sen[4], the slope and intercept were estimated. The choice of a linear trend was based on, in addition to the previous experiences of other software engineering researchers, the conservativeness of the linear function and the simplicity of its parameters. With this, the simulation model was completely defined.

Model validation. A crucial activity in simulation modeling is the validation phase[5, 6, 7]. It is an established practice to set aside a subset of the available data during model construction to be used later in the validation activity. In addition, the criteria presented in Table 2 are proposed for the validation of the model. Special emphasis was placed on the study of the variable "productivity" (defined as the ratio


U_X: NOISE DISTURBING REAL PROJECT SIZE X.
100% OBSERVATIONS
X - δX
X
δX
X + δX
DISTRIBUTION OF W
X - "REAL" PROJECT SIZE
W = X + U_X - "OBSERVED" PROJECT SIZE
$U_X \sim G$
WITH G ≡ U(- δX, + δX)
W IS UNIFORM IN THE INTERVAL
(X - δX X + δX)
I.E., 100% OF OBSERVATIONS ARE
EQUIPROBABLY DISTRIBUTED IN THIS
INTERVAL WITH δ = 0.1

(a)


E'Y: NOISE DISTRIBUTING REAL PROJECT EFFORT LN Y
σ
APPROX
70% (68.25%)
LNY - δLNY
LNY
LNY + δLNY
LNY
DISTRIBUTION OF LNY':
LNY = LOG OF "REAL" PROJECT EFFORT
LNY' = LNY + EY'
LOG O = "OBSERVED" PROJECT EFFORT
EY' ~ L≡N (0, σ^2) I.E.
EY' IS NORMAL WITH MEAN ZERO
AND STD DEV. σ = δ, INY, I.E.
APPROXIMATELY 70% OF THE TIME
LNY' FALLS WITHIN LNY ±0.2 LNY
ACCORDING TO GAUSSIAN PDF, δ = 0.2

(b)

Fig. 3. (a) Prior distribution of the noise U_x. (b) Prior distribution of the noise $E_{y'}$.

Table 2. Validation criteria for the simulation model

- Two-sample Kolmogorov-Smirnov GoF test
- One-sample Kolmogorov-Smirnov GoF test
- Nonparametric confidence intervals for
  - means
  - shape parameters
  - regression parameters
- Turing tests for
  - raw data plots
  - log/log plots
  - regression residual plots

of size to effort). This (output) variable was not directly generated during the simulation process but rather obtained as a synthesis of it. In addition to point estimation, confidence intervals were also considered when analyzing effort and productivity. Special attention was directed to the statistical distribution of the output variables. We refer to this latter criterion as ''validation by qualities,'' since in addition to the quantitative value of the parameters, we are interested in the way they manifest themselves. With this objective, the two-sample Kolmogorov-Smirnov (K-S) GoF test was applied to both the actual (DACS COBOL) data at hand and the corresponding simulated data. These two sets of values were examined for variable effort and productivity.

Additionally, a one-sample K-S GoF test was performed on the simulated data.

Also, for distributions like the gamma where the shape parameter determines its form, a nonparametric test of hypothesis and a confidence interval were obtained in order to observe whether their shape parameter was statistically larger (smaller) than a shape parameter equal to 1.

Finally, we studied the simulated data to see if it would also reproduce the same assumption violations and characteristics in the regression residual plots as the original DACS COBOL data. A Turing test for the plots is proposed. A summary table with the validation result follows (Tables 3A and 3B).

NUMERICAL EXAMPLE

A System Project Office (SPO) is supervising a new project. At the requirements analysis phase, advisors estimate that the project will attain a size of 40,000 lines of code (LOC). The SPO goes to the chart (Fig. 4A) and using the parametric regression line estimates the project effort to be 110 man-months (MM). If the SPO uses the nonparametric regression line, he/she will obtain a more conservative estimation of effort: 93 MM (exactly 93.74 MM for a project of 40,000 LOC).

However, the final project size is really unknown (a random variable), and the SPO has to deal with the fact of large variations in this initial estimate.

The simulation approach provides another alternative. In the example of Fig. 4B, 100 points have been generated. Suppressing the first 15 upper and lower points, there remain 70 pairs. These 70 remaining pairs define an empirical confidence region, and their density in the plot provides an indicator of the occurrence probability. Best and

Table 3A. Summary of validation scheme (I)

	Two-sample K-S GoF		One-sample K-S GoF		
Dataset	Effort	Productivity	Size	Effort	Productivity
1	X	X	X	X	X
2	X	X	X	X	0
3	X	X	X	X	X
4	X	X	X	X	X
5	X	X	X	0	0

X, pass—if 93.75% confidence interval covers parameter; 0, fail—if it does not cover parameter.

Table 3B. Summary of validation scheme (II)

				Symmetric noise
Distribution Free	93% Confidence Intervals	Means	Size	X
			Effort	0
			Product	X
		Shapes	Size	X
			Effort	0
			Product	X
		Regress Intercept	Parametric	X
			Nonparametric	X
		Regress slopes	Parametric	X
			Nonparametric	X

X, pass—if 93.75% confidence interval covers parameter; 0, fail—if it does not cover parameter.

worse cases can then be selected and analyzed in this context as indicated in Fig. 4B and Table 4.

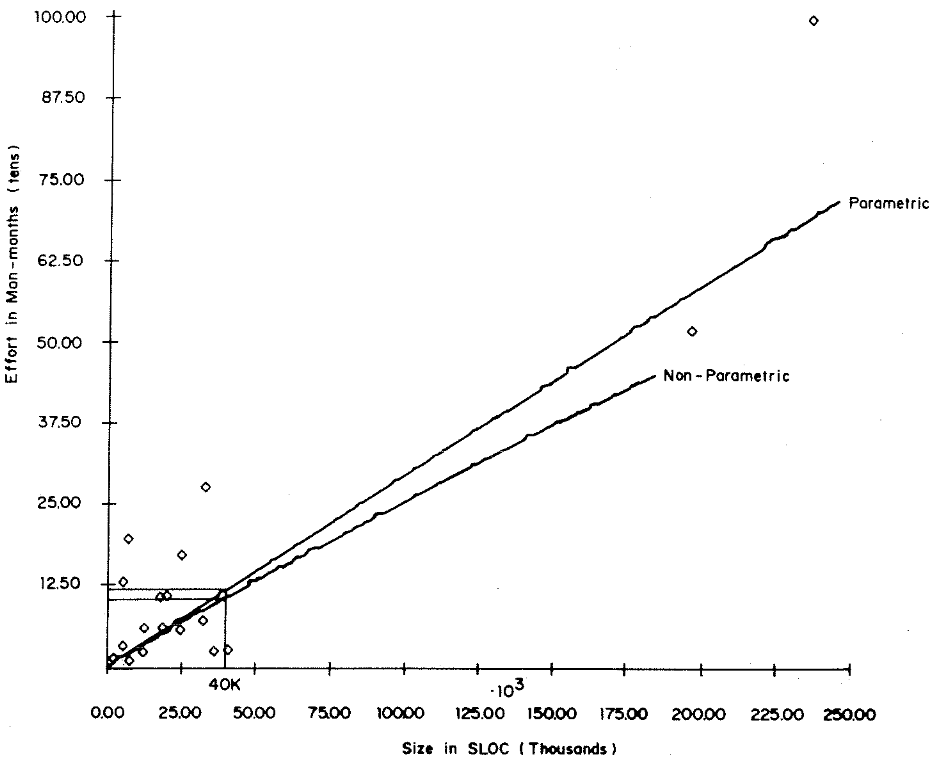
The objective of randomly generating a cloud of points is to derive an empirical confidence region. In general, this cloud of points will consist of a large number (i.e. 5000 or 10,000) of pairs (size, effort). A small program that counts the number of occurrences of these pairs per a predefined area and provides a grid of points to a tridimensional plotter of the "Surface II" type will be required. From these frequencies, the tridimensional plotter will provide either the (nomogram) contours or (tridimensional) surfaces. It is these frequency contours/surfaces that define the empirical confidence regions for the sensitivity analysis.

The selection of best and worst cases is based upon the additional information available for the same dataset to which this simulation model was fit. Every pair (size,

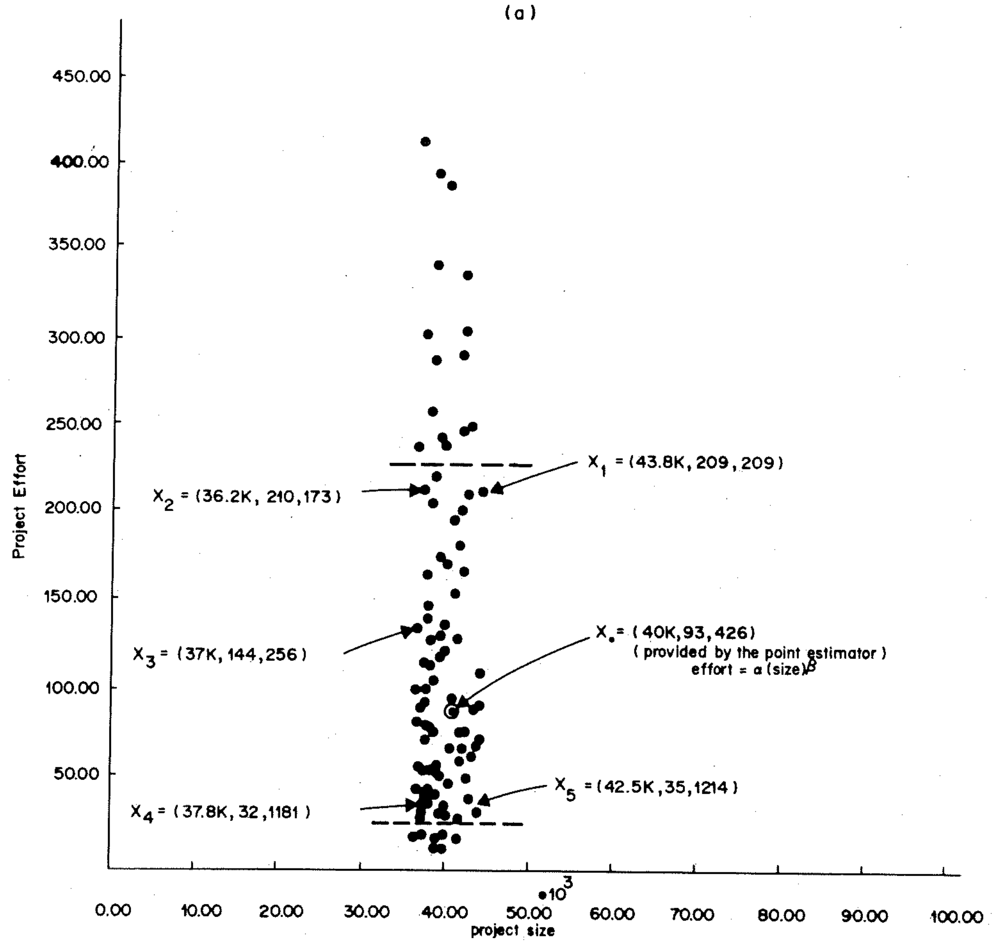
Table 4. Example of simulation results and sensitivity analysis

Alternative	Size ($\times 10^3$/LOC)	Effort (MM)	Productivity (LOC/MM)
X_0 (1)	40	93	426
X_1 (2)	43.8	209	209
X_2 (2)	36.5	210	173
X_3 (3)	37	144	256
X_4 (4)	37.8	32	1181
X_5 (4)	42.5	35	1214

1, nonparametric regression estimate; 2, worst cases; 3, inbetween cases; 4, best cases.



(a)



(b)

Fig. 4. (a) DACS COBOL projects—efforts vs size. Raw data: Parametric and nonparametric trends. (b) Example of sensitivity analysis using the simulation approach.

effort) represents a real project with its physical characteristics (i.e. development team experience, tools, application type, etc.). The process of "calibrating" this methodology to a given environment carries an implicit evaluation of the projects in the dataset.

The main contribution of the simulation methodology consists in providing these empirical confidence regions that have been generated using existing software engineering subjective prior knowledge. This prior knowledge is captured in the types of distributions defined for U_x and $\epsilon_{y'}$ and their parameters. From this simulation process, nomograms/plots are generated that provide the framework to identify where and with what frequency certain projects seem to cluster. From these plots, analysts can propose some reasonable best and worse cases with which to perform trade-off analysis. Furthermore, sensitivity analyses can also be performed by rerunning the simulator with the same prior distributions and other parameters, or with different priors, and comparing the different outcomes obtained. Finally, this framework provides an objective scenario in which different, but possible, alternatives can be examined and discussed[8].

CONCLUSIONS

A simulation approach has been presented, and a roadmap for its construction, validation and operation has been discussed and illustrated through an example using actual project data. The simulation model may provide an empirical confidence region for the pair (size, effort) and a confidence interval for productivity. These estimations may be obtained as early in the development of a software project as an initial estimate of the project size is available. These estimates are based upon the belief that existing software experience can be used to define a prior distribution describing the overall effect of undesired factors. They are very important and useful in the planning of resources (i.e. costs, staff and development time of a project). Finally, for those who deal with the high costs and uncertainties of software development and the difficulties of software management, the present tool is proposed as an independent and additional aid in their difficult task, capable of repaying its cost. A summary is provided in Table 5.

CAVEAT

The simulation model presented here as an illustration of the construction, validation and operation of the proposed approach should be evaluated within its own context. Originally, this model was implemented with restricted objectives. Therefore, consider the following three elements to properly evaluate it within the present example:

(i) The simulator was not originally developed for the use illustrated here. Model use determines model accuracy.

(ii) The distribution of the noise employed in this example is symmetrical. If the noise followed a nonsymmetrical distribution, or noises U_x and $\epsilon_{y'}$ were not independent, the usefulness of the simulator would increase even more.

(iii) The present approach holds independently of the function used to obtain effort from size. This function may well include other factors. In principle, it may also apply to more complex situations and to functions directly yielding software costs.

It was possible for the simulation model, with the limited resources and questionable

Table 5. Summary of the simulation approach

- Accurate estimates of software characteristics early in the project development are very important and very useful.
- Initial estimates are approximate.
- Software experience can be used to define a prior distribution that describes the overall random effects modifying these initial estimates.
- Using these prior distributions, software development characteristics can be simulated and empirical confidence regions for these characteristics obtained.
- Exact confidence intervals and confidence regions cannot be presently obtained through other available procedures.

data available, to achieve its restricted objectives. In this case, the proposed simulation approach, if extended with the required resources and data, may be a valid alternative worthy of repaying its cost.

Acknowledgment—I am indebted to Prof. Robert Sargent of the Department of IE/OR of Syracuse University for introducing me to the art of simulation and to Rocco F. Iuorno (IITRI) for many software engineering discussions and constant encouragement. Finally, the helpful suggestions of a reviewer improved the original manuscript to its final form. The Data and Analysis Center for Software is a Department of Defense Information Analysis Center operated by ITT Research Institute, Rome, New York, under contract no. F30602-83-C-0026, from the Air Force Systems Command, Rome Air Development Center, Griffiths Air Force Base, NY 13441.

REFERENCES

1. B. Boehm, Software engineering economics. *IEEE Trans. Software Engng.* **10,** 4–21 (1984).
2. S. Siegel, *Non-Parametric Statistics for the Behavioral Sciences.* McGraw-Hill, New York (1956).
3. E. L. Lehman, *Non-Parametric Statistical Methods Based on Ranks.* Holden-Day, San Francisco (1975).
4. P. K. Sen, Estimates of regression coefficient based on Kendall's tau. *JASA* **63,** 1379–1389 (1968).
5. R. Sargent, Verification and validation of simulation models. In *Progress in Modelling and Simulation,* pp. 159–169. Academic Press, New York (1981).
6. J. McLeod, editor, *Computer Modeling and Simulation.* Society for Computer Simulation, LaJolla (1982).
7. J. Emshoff & R. Sisson. *Design and Use of Computer Simulation Models.* Macmillan, New York (1970).
8. H. F. Martz and R. A. Waller, *Bayesian Reliability Analysis.* Wiley, New York (1982).
9. J. L. Romeu & S. Gloss-Soler, Some measurement problems detected in the analysis of software productivity data and their statistical consequences. Proceedings, COMPSAC83.