A Linear Combination Software Reliability Modeling Tool with a Graphically-Oriented User Interface

Allen P. Nikora
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099

Michael R. Lyu
Electrical and Computer Engineering Department
University of Iowa
Iowa City, IA 52242

Thomas M. Antczak
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099

Abstract

In our recent work, we have shown that forming linear combination of model results tends to yield more accurate predictions of software reliability. Using linear combinations also simplifies the practitioner's task of deciding which model or models to apply to a particular development effort. Currently, no commercially available tools permit such combinations to be formed within the environment provided by the tool.

Most software reliability modeling tools also do not take advantage of the high-resolution displays available today. Performing actions within the tool may be awkward, and the output of the tools may be understandable only to a specialist. We propose a software reliability modeling tool that allows users to formulate linear combination models, that can be operated by non- specialists, and that produces results in a form undetstandable by software developers and managements

1: Introduction

Over the past twenty years, many software reliability models have appeared in the literature [1]. Many of these models have been shown to be applicable to a sufficiently large number of failure data sets, so that development efforts would have some degree of confidence in using one or more of these models. Techniques for recalibrating models [31] and combining the results of models in a linear fashion [2,3,27,28,30] have been developed that appear to yield more accurate predictions than a single model. However, these models have not been used as widely as one might expect. A principal factor here is that it does not seem possible to make a priori determinations of which model or models will be the best suited to a particular development effort [1,2].

Another difficulty has been the lack of modeling tools that are easy for the non-specialist to use. For instance, many of the tools currently available were initially developed prior to the widespread availability of high-resolution displays, and therefore employ character-oriented user interfaces [4,5]. This characteristic of the tools may result in terse and cryptic command sequences, making it difficult for non-specialists or casual users to perform modeling actions with the tool. Since the results of the tools are displayed in a character-oriented fashion, the results will tend to be expressed in a way that is not easily understandable to non-specialists (e.g. model parameter values, tabular displays of interfailure times as opposed to failure rate curves). Considering the schedule pressures under which software developers and managers frequently operate, there is little incentive to learn how to operate a complicated new tool.

Also, the tools available today do not allow users to form linear combinations of model results within the tool environment. In earlier papers [2,3,27,28,30], we have shown that linear combinations of individual models can yield more accurate reliability predictions than the individual models themselves. To form linear combinations with current tools, the tool must be run several times to obtain the results from the desired component models of the combination. These results must then be combined in an application separate from the tool. Of course, this consumes more time than would be required if linear combinations could be formed within the tool environment.

In this paper we propose an architecture for a software reliability modeling tool that:

1. Supports the formation of linear combinations of model results within the tool environment.
2. Allows non-specialists to operate the tool and easily interpret the model results.

We refer to this tool as a Computer-Aided Software Reliability Estimation (CASRE) tool.

2: User Analysis

In developing the user interface for CASRE, it was necessary to identify the types of individuals that would use this tool as well as their software reliability knowledge. This knowledge determined the interaction

0-8186-2620-8/92 $3.00 © 1992 IEEE

style chosen for the user interface as well as the types of user interface objects that to be displayed. The six following types of users were identified:

1. Project managers
2. Line managers
3. Software development staff (system, software, and test engineers)
4. Software support staff (configuration management and product assurance personnel)
5. Consultants
6. Researchers

For each of these user categories, we describe their role in the software reliability measurement task, and further classify them according to schemes suggested by Sutcliffe and Schneiderman in their work on user interfaces [32,33] and summarized below.

Category	Definition and Value Ranges
User Knowledge	
task:	Knowledge of software reliability measurement techniques, rated as novice, skilled, or expert
computer:	Knowledge in use of computers to accomplish task, rated as novice, skilled, or expert
syntax:	Knowledge of syntax of actions required to accomplish task, rated as novice, skilled, or expert
Frequency:	How often involved in software reliability measurement task, measured as hourly, daily, weekly, monthly, or intermittent
Discretion:	Rated as compulsory or optional
Workload:	Proportion of time estimated to be dedicated to software reliability measurement - rated as low, medium, high
Interaction:	Data entry, low-level functions (e.g. synthesis of new combination models), high level functions (e.g. execution of one or more pre-specified models), all functions, uses output only.

2.1: Project Managers

Project Managers are typically former engineers who have made the transition to management. As such, they are familiar with basic techniques for interpreting statistical information, but may not be familiar with details of statistical modeling. These individuals typically receive reports generated by the support staff and use them as input to their decision making process. Consultants may also work with researchers to transfer academic findings to specific application domains.

Knowledge	
task:	skilled-
computer:	skilled-
syntax:	novice
Frequency:	monthly
Discretion:	optional
Workload:	low
Interaction:	Receives hardcopy reports. Rarely interacts directly with tool.

2.2: Line Managers

Line Managers are usually also former engineers who have made the transition to management. As with Project Managers, these individuals are familiar with the basic techniques for interpreting statistical information. Line Managers receive reports from their support staff and use them as input to their decision process. Since Line Managers are usually closer to the actual development effort, they would tend to request reports more frequently than Project Managers. They may also use some of the basic tool capabilities (e.g. running pre-specified models, but not creating new ones).

Knowledge	
task:	skilled-
computer:	skilled-
syntax:	novice
Frequency:	biweekly
Discretion:	optional
Workload:	low
Interaction:	Receives hardcopy reports. Occasional use of high-level functions of tool.

2.3: Development Staff

Users of software reliability measurement techniques within the development organization include system engineers, software engineers, programmers, and test engineers. These individuals typically have degrees in technical disciplines, extensive software development experience, and some additional training in the methods and tools that apply to their assignment. These individuals use modern software development tools on a regular basis. They will be familiar with the basics of probability theory and statistics, and may have advanced training in statistical modeling techniques. Currently, however, they rarely have had training in software reliability theory, methods, or tools.

Knowledge	
task:	skilled-
computer:	expert
syntax:	expert
Frequency:	weekly

Discretion:	compulsory, subject to Project or Line management policy
Workload:	low
Interaction:	Use of high-level functions of tool.

2.4: Support Staff

Users of software reliability measurement techniques within the support staff include configuration management specialists and quality assurance personnel. These individuals include both clerical and technical personnel. Most of these individuals have extensive experience in configuration management and quality assurance activities across a wide range of projects. Consequently, some support staff members have training or experience with software reliability measurement techniques at various levels.

Knowledge	
task:	novice+
computer:	skilled
syntax:	skilled
Frequency:	weekly
Discretion:	compulsory, subject to Project or Line management policy
Workload:	low
Interaction:	Primarily high-level functions, some use of low-level functions.

2.5: Consultants

A software reliability consultant typically has an advanced degree in a technical discipline extensive background in all aspects of software reliability measurement, and significant software development experience. This individual plays a key role in introducing software reliability measurement techniques into a project at all levels. This includes assisting the Project and Line Managers in setting software reliability goals and interpreting results, and assisting the development and support staffs in selecting and using models and support tools.

Knowledge	
task:	expert-
computer:	expert
syntax:	expert
Frequency:	inter-mittent
Discretion:	optional
Workload:	high
Interaction:	All functions.

2.6: Researchers

Researchers are typically members of the faculty at a university who develop or refine reliability models. Researchers may work with consultants in transferring knowledge from the academic to environment to specific applications domains.

Knowledge	
task:	expert
computer:	skilled
syntax:	expert
Frequency:	daily
Discretion:	optional
Workload:	high
Interaction:	All functions.

2.7: User Analysis Summary and Recommendations

Table 1 summarizes the user analysis given above. We see from Table 1 that the reliability measurement task is performed within a software development effort on, at best, a weekly basis (discounting the time that may have been spent with a consultant in setting up a reliability measurement program). Also, some of the users performing the task the most frequently have the lowest level of reliability measurement knowledge. Given these factors, the goals of low learning time and good retention over time were the primary concerns in designing the CASRE interface. These findings suggest that a menu-oriented or direct manipulation style of interaction, or perhaps a combination of the two, is appropriate.

	User Knowledge						
	Task	Computer	Syntax	Frequency	Discretion	Work-load	Interaction
Project Manager	skilled-	skilled-	novice	monthly	optional	low	hardcopy
Line Manager	skilled-	skilled-	novice	weekly	optional	low	hardcopy, some high level functions
Development Staff	skilled-	expert	expert	weekly	compulsory	low+	high level functions
Support Staff	novice+	skilled	skilled	weekly	compulsory	low+	high level functions, some low level functions
Consultant	expert-	expert	expert	intrmtnt	optional	high	all functions
Researcher	expert	skilled	expert	daily	optional	high	all functions

Table 1 - Summary User Profiles

While important, good speed of performance was not deemed as critical as the other two goals. When running a complicated model, such as the Littlewood-Verrall model [19], on a set of failure data, it is to be expected that results will not be immediately available. We therefore specified the goal that the throughput of the modeling section should at least be comparable to that of some of the more popular tools currently in use.

3: The CASRE Tool - High-Level Structure and Functionality

This section describes the high-level architecture and the basic functionality of the CASRE tool. To implement the recommendations resulting from the user analysis, it is planned to implement CASRE on top of a windowing system (e.g. X-Windows/MOTIF, DOS Windows 3.0). Figure 1 shows the proposed high-level architecture for CASRE, whose major functional areas are:

Data Modification
Failure Data Analysis
Modeling and Measurement
Modeling/Measurement Results Display

Much of CASRE's functionality is available in current software reliability tools [4,5]. However, a feature unique to CASRE allows users to combine the results of several models in addition to executing a single model. Feedback from the Model Evaluation block assists users in identifying a model or combination of models best suited to the failure data being analyzed. Moreover, the I/O facilities, the user interface, and the measurement procedures are greatly enhanced in this tool.

3.1: Data Modification

CASRE allows users to create new failure data files, modify existing files, and perform global operations on files. Editing CASRE allows users to create or alter failure history data files. A simplified spreadsheet-like user interface allows users to enter time between failures or test interval lengths and failure counts from the keyboard. Users are also allowed to invoke a preferred editor (e.g. emacs or vi).

3.2: Smoothing

Since input data to the models is often fairly noisy, the following smoothing techniques are proposed:

- Sliding rectangular window
- Hann window
- General polynomial fit
- Cardinal Spline
- Specific cubic-polynomial fits (e.g. B-Spline, Bezier Curve)

Users select smoothing techniques appropriate to the failure data being analyzed. The smoothed input data can be plotted, used as input to a reliability model, or written out to a new file for later use. Summary statistics for the smoothed data can also be displayed (see "Failure Data Analysis" below).

3.3: Data Transformation

In some situations, logarithmic, exponential, or linear transformations of the failure data produce better or more understandable results. The following operations, currently available in some tools, allow users to transform an entire set of failure data in this manner.

- log(a * x(i)) + b); x(i) represents a failure data item, and a and b are user-selectable scale factors
- exp(a * x(i) + b)
- x(i) ** a
- x(i) + a
- x(i) * a
- User-specified transformations might also be allowed.

As with smoothing, users select a specific transformation. Users are able to manipulate transformed data as they would smoothed data.

3.4: Failure Data Analysis

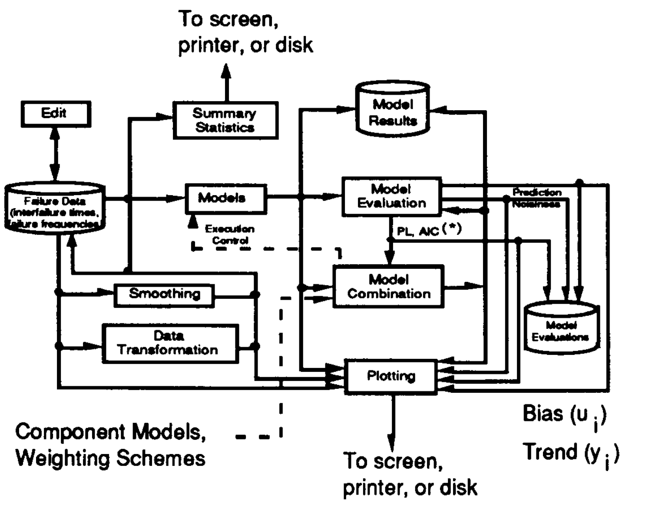
The "Summary Statistics" block in Figure 1 allows users to display the failure data's summary statistics, including the mean and median of the failure data, 25% and 75% hinge points, skewness, and kurtosis [6].

3.5: Modeling and Measurement - Single Model Execution

Figure 1 shows two modeling functions. The "Models" block executes single software reliability models on a set of failure data. The "Model Combination" block allows users to execute several models on the failure data and combine the results of those models. We include this capability because our experience in combining the results of more than one model indicates that such "combination models" may provide more accurate reliability predictions than single models [3]. The block labeled "Model Evaluation" allows users to determine the applicability of a model to a set of failure data.

Based on our experience in applying software reliability models, we include the following models in CASRE:

1) Bayesian Jelinski-Moranda (BJM) [7,8,9]
2) Goel-Okumoto Model (GO) [13]
3) Jelinski-Moranda Model (JM) [14,15]
4) Keiller-Littlewood Model (KL) [16,17]
5) Littlewood Model (LM) [18]
6) Littlewood NHPP Model (LNHPP) [1]
7) Littlewood-Verrall Model (LV) [19]
8) Musa-Okumoto Model (MO) [20]
9) Generalized Poisson Model (PM) [10]
10) Schneidewind Model (SM) [21]
11) Yamada S-Shaped Model (YM) [22]
12) Geometric Model (GM) [10]
13) Duane Model (DU) [11,12]



(*) PL = Prequential Likelihood, AIC = Akaike Information Criterion

Figure 1: High-Level Architecture for CASRE

The models should be implemented to allow input to be in the form of interfailure times or failure frequencies. CASRE allows users to choose the parameter estimation method (maximum likelihood, least squares, or method of moments). Model outputs include:

- Current estimates of failure rate or inter-failure time
- Current estimates of reliability
- Model parameter values, including high and low parameter values for a user-selectable confidence bound
- Current values of the probability density function and cumulative density function associated with the observed interfailure times or failure frequencies
- u-plots and y-plots [1,34] indicating bias and trends in the bias of the model

Users can display these quantities on-screen or write them to disk.

3.6: Combination Models

CASRE allows users to combine the results of several models according to the Equally-Weighted Linear Combination (ELC), Median- Oriented Linear Combination (MLC), Unequally-Weighted Linear Combination (ULC), or Dynamically-Weighted Linear Combination (DLC) schemes described in [3,27]. Users may also be allowed to define their own weighting schemes. The resulting combination models could be further used as the component models to form another combination model.

3.7: Model Evaluation

CASRE includes the following statistical methods to help users determine the applicability of a model (including "combination models") to a specific failure data set:

- Computation of prequential likelihood (PL) function (the "Accuracy" criterion).
- Determination of the probability integral transform u_i [1,34], (plotted as the u-plot - the "Bias" criterion).
- Computation of the normalized logarithmic transform of u_i, y_i [1,34], to produce the y-plot (the "Trend" criterion).
- Noisiness of model predictions (the "Noise" criterion).
- The Akaike Information Criterion (AIC) [23], similar in concept to prequential likelihood, could also be implemented.

This model evaluation function would also compute goodness-of-fit measures (e.g. Chi-Square test). The PL and AIC outputs are used as input to "Model Combination" to determine the relative contribution of individual models if the user has specified a combination model.

3.8: Display of Results

CASRE graphically displays model results in the following forms:

- Interfailure times/failure frequencies, actual and estimated
- Cumulative failures, actual and estimated
- Reliability growth, actual and estimated

Actual and estimated quantities are available on the same plot. Plots include user-specified confidence limits. Users are able to control the range of data to be plotted as well as the usual cosmetic aspects of the plot (e.g. X and Y scaling, titles). In a windowing environment, multiple plots could be simultaneously displayed. CASRE allows users to save plots displayed on-screen as a disk file or to print them. One public-domain tool, version IV of SMERFS [4], can write the data used to produce a plot to a file that can be imported by a spreadsheet, a DBMS, or a statistics package for further analysis. CASRE also includes this capability. The plotting function also produces u-plots and y-plots from Model Evaluation's u_i and y_i outputs. These plots indicate the degree and direction of model bias and the way in which the bias changes over time.

4: Application Procedure

Figures 2-8 show a series of screen dumps for the described CASRE tool, using simulated failure data. It can be seen that the application of models to failure data is a straightforward process. The user is also given a considerable amount of choice in the models to be applied, making it easy to identify an appropriate model for a particular development.

4.1: Opening a Failure Data File

The screen is shown in Figure 2. To choose a set of failure data on which a reliability model will be run, the user selects the "File" menu with the mouse. After selecting the "Open" option in the File menu, a dialogue box for selecting a file appears on the screen. The current directory appears in the editable text window at the top of the dialogue box. The failure history files in that directory are listed in the scrolling text window. The user selects a file by highlighting its name (scrolling the file name window if necessary) and then pressing the "Open" button. To change the current directory, the user enters the name of the new directory in the "Current Directory" window and presses the "Change Directory" button. Pressing the "Cancel" button removes the dialogue box from the screen.

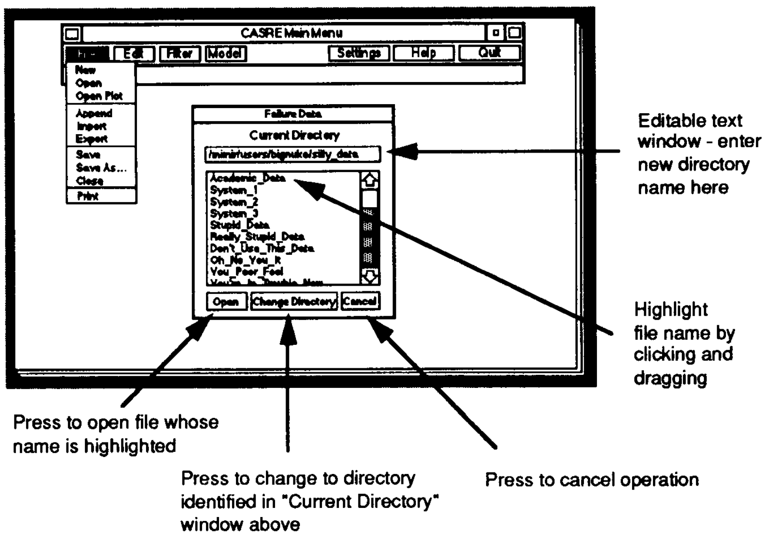


Figure 2 - Failure Data Selection

4.2: Preliminary Failure Data Analysis

The screen is shown in Figure 3. After opening a failure history file, the contents of the file are displayed in tabular and graphic forms. The tabular representation resembles a spreadsheet, and the user can perform similar types of operations (e.g. selecting a range of data, deleting one or more rows of data). All of the fields can be changed by the user except for the "Interval Number" field (or "Error Number" field if the data is interfailure times). In this example, the selected data set is in the form of test interval lengths and number of failures per test interval. The user can scroll up and down through this tabular representation and resize it as per the MOTIF or DOS Windows conventions. The large graphics window displays the same data as the worksheet. If the failure data set is interfailure times, the initial graphical display is interfailure times. If, as in this example, the failure data set is test interval lengths and failure counts, the initial graphical display is the number of failures per test interval. The display type can be changed by selecting one of the items from the "Display Type" menu associated with the graphics window. The user can move forward and backward through the data set by pressing the right arrow or left arrow buttons at the bottom of the graphics window. Finally, the iconified window at the lower left corner of the screen lists the summary statistics for the data. To open this window, the user clicks on the icon. The following information is then displayed in a separate window:

- Number of observations in this data set
- Type of observations made (interfailure times or test interval lengths and failure counts)
- Mean value of the observations
- Minimum and maximum values
- Median
- 25% and 75% hinges
- Standard deviation and variance
- Skewness and Kurtosis

4.3: Failure Data Selection and Editing

The screen is shown in Figure 4. The user will frequently use only a portion of the data set to estimate the current reliability of the software. This is because testing methods may change during the testing effort, or different portions of the data set may represent failures in different portions of the software. To use only a subset of the selected data set, the user may simply "click and drag" on the tabular representation of the data set to highlight a specific range of observations. The user may also select previously-defined data ranges. To do this, the user chooses the "Select Range" option of the Edit menu. This brings up a dialogue box containing a scrolling text window in which the names of previously-defined data ranges and the points they represent are listed. To select a particular range, the user highlights the name of the range in the scrolling text window and presses the "OK" button. Pressing the "Cancel" button removes the dialogue box and the Edit menu from the screen. Once a range has been selected, all future modeling operations will be only for that range. The selected data range is highlighted in the tabular representation. The graphics display will change to include only the highlighted data range. All other observations will be removed from the graphics display.

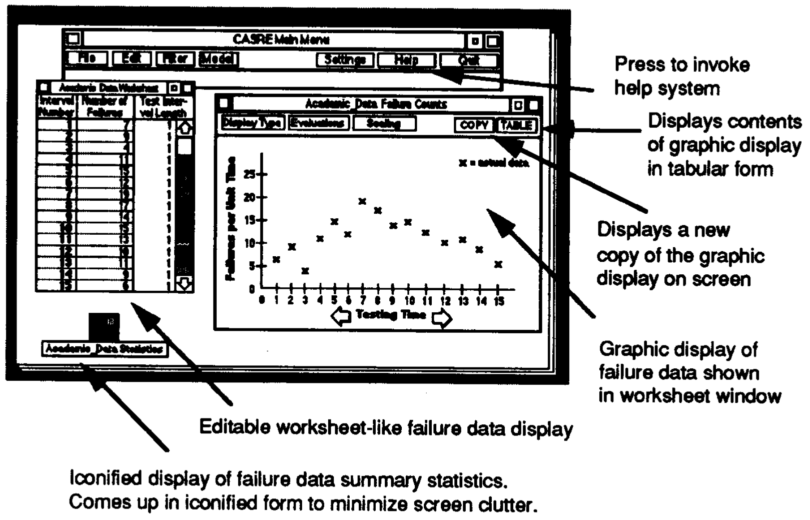

Press to invoke help system
Displays contents of graphic display in tabular form
Displays a new copy of the graphic display on screen
Graphic display of failure data shown in worksheet window
Editable worksheet-like failure data display
Iconified display of failure data summary statistics. Comes up in iconified form to minimize screen clutter.

Figure 3 - Initial Failure Data Display

4.4: Data Filtering

The screen is shown in Figure 5. After selecting a data range, the user may wish to transform the file or smooth the data. Software failure data is frequently very noisy; smoothing the data or otherwise transforming it may improve the modeling results. To do this, the user selects one of the options in the "Filter" menu. There are five affine transformations which the user may apply to the data, and six types of smoothing. Transformations and smoothing operations may be pipelined - for example, the user could select the "ln(A * X(i) + B)" transformation followed by the B-spline smoothing operation. The number of filters that may be pipelined is limited only by the amount of available memory. The tabular representation of the failure is changed to reflect the filter, as is the graphical display of the data. The type of filter applied to the data is listed at the right hand edge of the graphics display window. In this example, we have applied a B spline to the data. Once a series of filters has been applied to the data, the user may remove the effect of the most recent filter by selecting the "Undo" option of the Filter menu. To remove the effect of the entire series of filters, the user selects the "Undo All Filters" option of the Filter menu.

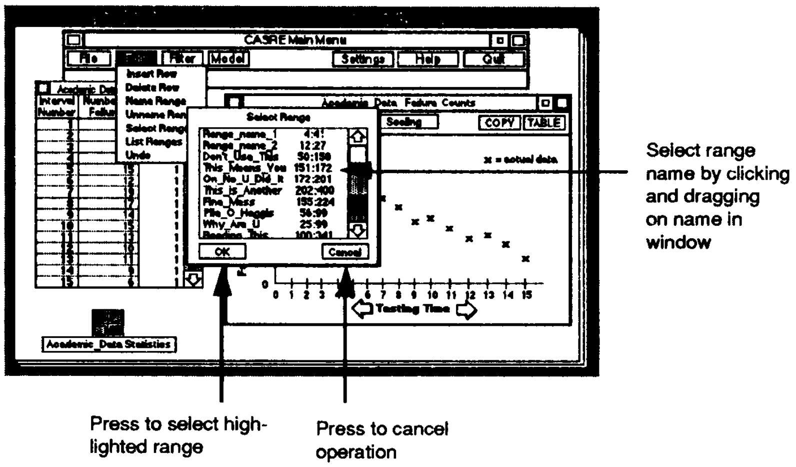


Figure 4 - Subsetting Failure Data

4.5: Applying Software Reliability Models

The screen is shown in Figure 6. After the user has opened a file, selected a data range, and done any smoothing or other transformation of the data, a software reliability model can be run on the data. In the Model menu, the user has the choice of 13 individual models or a set of models which combine the results of two or more of the individual models. The user may also choose the method of parameter estimation (maximum likelihood, least squares, or method of moments), the confidence bounds that will be calculated for the selected model, and the interval of time over which predictions of future failure behavior will be made.

4.6: Selecting the Best Model(s)

The screen is shown in Figure 7. There are many models from which to choose in this tool. The user may not know which model is most appropriate for the data set being analyzed. Using CASRE, the user can request, "display the results of the individual model which best meets the four prioritized criteria of accuracy (based on prequential likelihood), biasedness, trend, and noisiness of prediction." To do this, the user first selects the "Individual" option of the Model menu. A submenu then appears, on which 13 individual models are listed, as well as a "Choose Best" option. The user selects the "Choose Best" option, which results in a "Selection Criteria" dialogue box being displayed. The user moves the four sliders in this dialogue box back and forth to establish the relative priorities of the four criteria. Numerical values of the priorities are displayed in the text boxes on the right side of the dialogue box. Once the priorities have been established, the user presses the "OK" button. CASRE then proceeds to run all of the individual models against the data set, first warning the user that this is a time-consuming operation and allowing cancellation of the operation. If the user continues, CASRE provides the opportunity for cancellation at any time if the user decides that the operation is taking too much time.

4.7: Displaying the Final Results

The screen is shown in Figure 8. Once a model has been run on the failure data, the results are graphically displayed. Actual and predicted data points are shown, as are confidence bounds. The model is identified in the window's title bar; the percent confidence bounds are given at the right side of the graphics window. This concludes one round of software reliability measurement

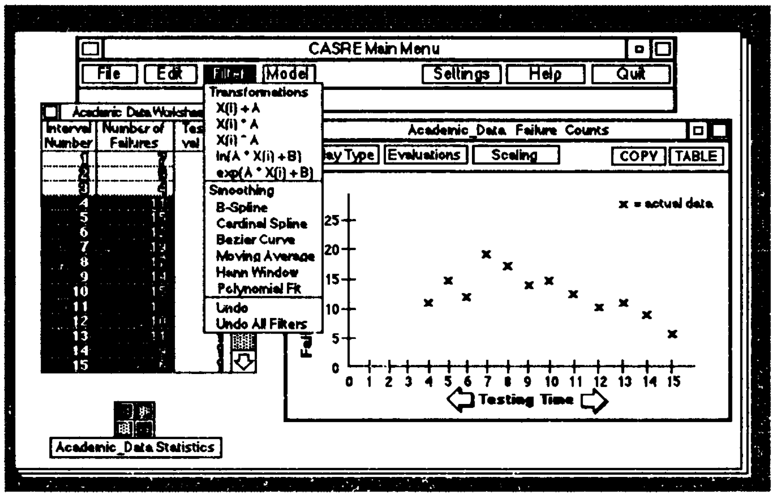
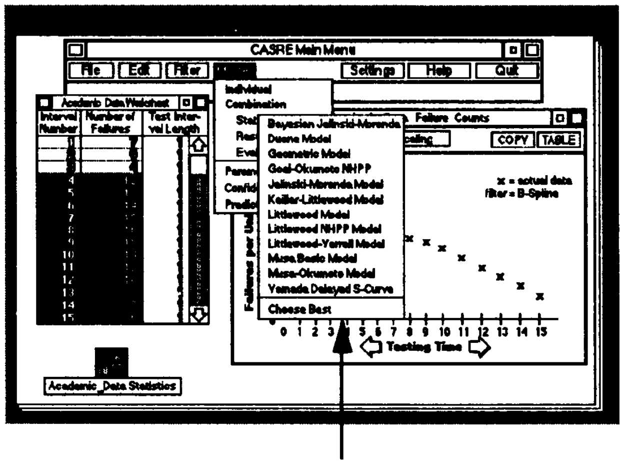


Figure 5 - Filtering the Failure Data



Press to have the tool select the model that best fits a set of user-specified criteria.

Figure 6 - Model Application

with CASRE.

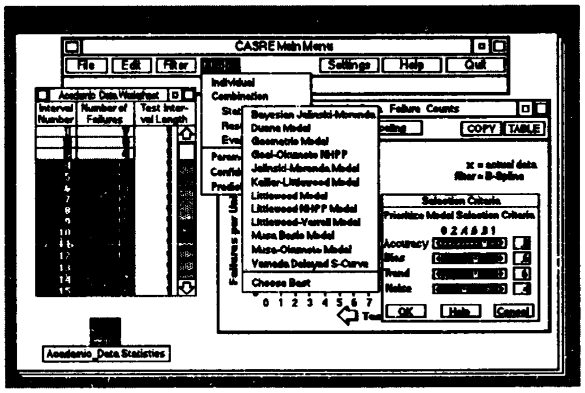
5: General Experiences and On-Going Work

A Hypercard prototype of the CASRE interface was first presented and demonstrated at the 14th Minnowbrook Workshop on Software Engineering [30]. Remarkably, there were no suggestions for change that would have meant any significant re-organization of the tool. Currently, the Air Force Operational and Test Center (AFOTEC) is funding the implementation of this tool for a Microsoft Windows 3.0 environment. The modeling capability of CASRE will be based on the mathematical library of SMERFS version IV. We decided that this would be the most effective way of accomplishing the task within the allocated resources. Rather than writing a new set of modeling routines, it made more sense to make use of an already existing modeling library that had been extensively tested. Implementation of the linear combination modeling facility will be a straightforward task, since all that is needed is a control mechanism to sequence through the selected models and assign weights to the results of individual models.

Since the time of the Minnewbrook presentation, some changes to the original concept have been made. The most significant change is in the model selection and application area (illustrated in Figures 6 and 7). Recall from the previous section that to execute a model, users would choose a model from a sub-menu of the Model pull-down menu. This would have resulted in a sub-menu for individual models and a set of control panels and sub-menus for the linear combination models. To run more than one model would be a tedious exercise with this type of interaction; users would have to choose one model, wait for it to complete, then choose the next model, and so forth.

A discussion with the AFOTEC sponsors revealed that a more sensible model selection and execution mechanism would be a checklist, in which users would indicate all of the models, individual or combination, to be executed during a modeling run. Upon completing the checklist, users would select an item on the checklist that would start execution of the models. This would free users to perform other tasks while the models were executing. As with other applications involving possibly lengthy computations, users would be given the option to terminate execution of the chosen set of models at any time.

This change has led to modifications in the drawing window in which modeling results are displayed. As originally conceived, this window would display the raw data and the results of only one model. Now that the user will be allowed to execute more than one model at a time, the drawing window will change to allow users to specify which models' results will be displayed in the window. This will be accomplished by a checklist similar to that used to specify models for execution. However, in this "display selection" checklist, only the models that have been executed will be listed. This facility will allow users to



To specify the criteria by which a model will be judged "best", the user moves the slide bars on the "Selection Criteria" control panel at the right edge of the screen to set the relative weights of four criteria.

Figure 7 - Weighting Model Selection Criteria

easily compare the outputs, and hence the behavior, of two or more models.

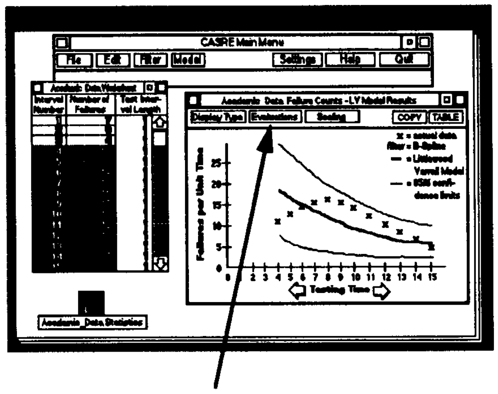


Figure 8 - Model Results Display

6: Conclusions and Future Work

We have proposed a set of linear-combination models for more accurate measurement of software reliability. These models have shown promising results when compared with the traditional single-model approaches. To relieve the tedious work involved in applying these approaches, a CASE tool, called CASRE, is proposed to automate the software reliability measurement task. For the purpose of model validation and determining tool applicability, we need to obtain enough data to compare software reliability models and predictions across various types of software projects. Some data sets can be found in [24], [25], [26], and [29]. In future investigations, we will apply more data sets to the proposed combination models for the purpose of validating them, and for refining the structure and functionality of the CASRE tool. Interim versions of the CASRE tool will be prepared and refined; potential users will be identified and asked to use these interim versions and evaluate them based on their experience within a software development effort. These evaluations will be used in refining the structure and functionality of the tool.

Acknowledgements

The research described in this paper was carried out at the University of Iowa under a faculty starting fund, and at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration, through the JPL Director's Discretionary Fund. Support for the implementation of the tool described herein is provided by the Air Force Operational and Test Evaluation Center, under Task Order RE-182, Amendment 655, Proposal No. 80-3417. Modifications to the SMERFS modeling library to support the model evaluation methods discussed in this paper will be accomplished by Dr. W. H. Farr of the Naval Surface Warfare Center, Dahlgren, VA.

References

1. A. A. Abdel-Ghaly, P. Y. Chan, B. Littlewood, "Evaluation of Competing Software Reliability Predictions," IEEE Transactions on Software Engineering, vol. SE-12, pp.950-967, September, 1986.

2. M. R. Lyu, "Measuring Reliability of Embedded Software: An Empirical Study with JPL Project Data," in Proceedings of the International Conference on Probabilistic Safety Assessment and Management, Beverly Hills, California, February, 1991.

3. M. R. Lyu, A. Nikora, "Software Reliability Measurements Through Combination Models: Approaches, Results, and a Case Tool," in Proceedings of the 15th Annual International Computer Software and Applications Conference (COMPSAC91), Tokyo, Japan, September, 1991.

4. W. H. Farr, O. D. Smith, "Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) User's Guide," TR84-373, Revision 1, NavSWC, December, 1988.

5. B. Littlewood, A. A. Abdel-Ghaly, P. Y. Chan, "Tools for the Analysis of the Accuracy of Software Reliability Predictions," in Software System Design Methods, pp.299-335, Springer-Verlag, Heidelberg, 1986.

6. R. V. Hogg, A. T. Craig, Introduction to Mathematical Statistics, MacMillan, New York, 1978

7. H. Joe, N. Reid, "Estimating the Number of Faults in a System," Journal of the American Statistics Association, vol, 80, pp. 222-226, March, 1985.

8. N. Langberg, N. D. Singpurwalla, "A Unification of Some Software Reliability Models via the Bayesian Approach," Technical Report TM-66571, George Washington University, 1981.

9. B. Littlewood, A. Sofer, "A Bayesian Modification to the Jelinski-Moranda Software Reliability Model," Software Engineering Journal, vol. 2, pp. 30-41, 1987.

10. W. H. Farr, "A Survey of Software Reliability Modeling and Estimation," Technical Report 82-171, NavSWC, 1983.

11. J. T. Duane, "Learning Curve Approach to Reliability Modeling," IEEE Transactions on Aerospace, vol. AS-2, pp. 563- 566, 1984.

12. L. H. Crow, "Confidence Interval Procedures for Relaibility Growth Analysis," Technical Report 197, U. S. Army Material Systems Analysis Activity, Aberdeen, Maryland, 1977.

13. A. L. Goel, K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," IEEE Transactions on Reliability, vol. R-28, pp. 206-211, 1979.

14. Z. Jelinski, P. B. Moranda, "Software Reliability Research," in Statistical Computer Performance Evaluation, ed. W. Freiberber, pp. 465-484, Academic Press, New York, 1972.

15. M. Shooman, "Operational Testing and Software Reliability During Program Development," in Proceedings of the 1973 Symposium on Computer Software Reliability, pp. 51-57, New York, April, 1973.

16. P. A. Keiller, B. Littlewood, D. R. Miller, A. Sofer, "Comparison of Software Reliability Predictions," in Proceedings of the 13th International Symposium on Fault-Tolerant Computing, pp. 128-134, 1983.

17. P. A. Keiller, B. Littlewood, D. R. Miller, A. Sofer, "On the Quality of Software Reliability Predictions," in Proceedings of the NATO ASI Electronic Systems Effectiveness and Life Cycle Costing, pp. 441-460, Berlin, Springer, Norwich, England, 1983.

18. B. Littlewood, "Stochastic Reliability Growth: A Model for Fault-Removal in Computer Programs and Hardware Designs," IEEE Transactions on Reliability, vol. R-30, pp.313-320, October, 1981.

19. B. Littlewood, J. J. Verrall, "A Bayesian Reliability Growth Model for Computer Software," Journal of the Royal Statistics Society C, vol. 22, pp. 332-346, 1973.

20. J. D. Musa, K. Okumoto, "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement," in Proceedings of the Seventh International Conference on Software Engineering, pp. 230-238, Orlando, Florida, 1984.

21. N. F. Schneidewind, "Analysis of Error Processes in Computer Software," in Proceedings of the International Conference on Reliable Software, pp. 337-346, Los Angeles, 1975.

22. S. Yamada, M. Ohba, S. Osaki, "S-Shaped Reliabiltiy Growth Modeling for Software Error Detection," IEEE Transactions on Reliability, vol. R-32, pp. 475-478, December, 1983.

23. H. Akaike, "A New Look at Statistical Model Identification," IEEE Transactions on Automatic Control, vol. AC-19, pp. 716-723, 1974.

24. R. Troy, Y. Romain, "A Statistical Methodology for the Study of the Software Failure Process and its Application to the ARGOS Center," IEEE Transactions on Software Engineering, vol SE-12, no. 9, pp. 968-978, September, 1986.

25. W. Erlich, J. Stampfel, J. Wu, "Application of Software Reliabiltiy Modeling to the Product Quality and Test Process," in Proceedings of the 12th International Conference on Software Engineering, Nice France, March, 1990.

26. K. C. Zinnel, "Using Software Reliability Growth Models to Guide Release Decisions," in Proceedings of the 1990 Internation Symposium on Software Reliability Engineering, Washington, DC, April, 1990.

27. M. R. Lyu, A. Nikora, "A Heuristic Approach for Software Reliability Prediction: the Equally-Weighted Linear Combination Model," Proceedings of the 1991 International Symposium on Software Reliability Engineering, pp. 172-181, Austin, Texas, May, 1991.

28. M. R. Lyu, A. Nikora, "An Empirical Approach for Software Reliability Measurement by Linear Combination Models," submitted to Special Issue of IEEE Software, July, 1992.

29. J. D. Musa, "Software Reliability Data," RADC Technical Report, 173 pp., DACS, Rome Air Development Center, 1980.

30. A. P. Nikora, M. R. Lyu, T. M. Antczak, W. H. Farr, "Linear Combination Software Reliability Models and a Proposed Reliability Modeling Tool," 14th Minnowbrook Workshop on Software Engineering, Blue Mountain Lake, New York, July, 1991.

31. S. Brocklehurst, P. Y. Chan, B. Littlewood, J. Snell, "Recalibrating Software Reliability Models," IEEE Transactions on Software Engineering, vol, SE-16, no. 4, pp. 458-470, April, 1990.

32. A. Sutcliffe, Human-Computer Interface Design, Springer-Verlag, 1989.

33. B. Shneiderman, Designing the User Interface, Addison-Wesley, 1987.

34. J. D. Musa, A. Iannino, K. Okumoto, Software Reliability: Measurement, Prediction, Application, McGraw-Hill Book Company, New York, New York, 1987.