An Open Source Software Reliability Tool A Guide for Users

Vidhyashree Nagaraju[1], Karthik Katipally[1], Richard Muri[1], Lance Fiondella[1], and Thierry Wandji[2]
[1]Electrical and Computer Engineering, University of Massachusetts Dartmouth, MA, USA
[2]Naval Air Systems Command, Patuxent River, MD, USA
Email: {vnagaraju,kkatipally,lfiondella}@umassd.edu,ketchiozo.wandji@navy.mil

***Abstract*—This paper presents an open source software reliability tool to automatically apply methods from software reliability engineering, including visualization of failure data, application of software reliability growth models, inferences made possible by these models, and assessment of goodness of fit. The application and source code are available through the web. The open source nature of this tool will enable unprecedented levels of collaboration among researchers and practitioners from industry and government within a single shared platform.**

I. Software Reliability Engineering

Software reliability engineering [1] is an established field, which is over 40 years old. However, research has not reached it target audience, namely software and reliability engineers. Often, these audiences are reluctant to apply software reliability models because they lack either the knowledge of the underlying mathematics or the time to develop expertise and implement models in their work. Automated tools are needed to ensure software reliability engineering methods are utilized.

Previous automated tools for software reliability engineering include: SMERFS (Statistical Modeling and Estimation of Reliability Functions for Software) [2] which was incorporated into CASRE (Computer-Aided Software Reliability Estimation tool) [3] and SRATS (Software Reliability Assessment Tool on Spreadsheet) [4]. Although popular among the user community, the primary shortcoming of these previous tools is that they were implemented as desktop applications by a single researcher, which limited the models incorporated to those the individual researcher was familiar with.

To overcome the limitation of previous software reliability tools, this paper presents an open source software reliability tool (SRT) to promote collaboration among members of the international software reliability research community and users from industry and government organizations. The application is implemented in the R programming language, an open source environment for statistical computing and graphics. It also accessible through a web-enabled framework web at *sasdlc.org* (System and software development life cycle). The code is accessible through the GitHub repository at *github.com/lfiondella/SRT*. Its architecture will enable incorporation of existing software reliability models into a single tool, enabling more systematic comparison of models than ever before. Presently, interfailure time, failure time, and failure count data formats are supported. Implemented functionality includes: two trend tests for reliability growth, two failure rate, Jelinski-Moranda [1] and geometric [1], and three failure counting models, Goel-Okumoto [1], [5], delayed S-shaped [6], and Weibull [7] models as well as two measures of goodness of fit. Inferences such as the time to achieve a target reliability or detect a specified number of additional failures have also been implemented.

Section II provides a brief overview of the tool's user interface, which is divided into four primary activities, including selection, analysis, and filtering Data II-A, model set up an application II-B, querying of model results II-C, and model evaluation II-D. Section III concludes the paper and identifies future work.

II. Software reliability tool (SRT)

Figure 1 shows the four primary tabs of the Shiny interface of the open source software reliability tool.

Fig. 1: Main tab interface

A. Tab 1: Select, Analyze, and Filter Data

Figure 2 shows the options in the first tab, including file selection, data visualization, and trend test analysis. User can begin to use the tool by specifying the input file as either an Excel spreadsheet (.xlsx) or a CSV (comma separated value) (.csv) format. The input file must follow the format shown in Figure 3, where (FN) indicates failure number, (IF) interfailure time, and (FT) failure time.

Clicking on the **Choose File** button enables the user to browse and upload the input data file. The progress bar message "Upload complete" indicates a successful file upload. By default, a plot of the cumulative failures for the uploaded data set is displayed; Figure 4 shows the SYS1 data set [1].

Below the progress bar, the user is able to choose a sheet from a dropdown menu listing the data sets. csv files can contain only one data set, while Excel files can contain one data set per sheet. Data sets that do not comply with the input format will not be available in the dropdown menu. During upload, the tool converts data into failure times, failure counts, and inter failure data formats regardless of the input format so that any model can be applied to any data set. 31 data sets were taken from the Handbook of Software Reliability Engineering [1] and prepared in the file format. Of these,

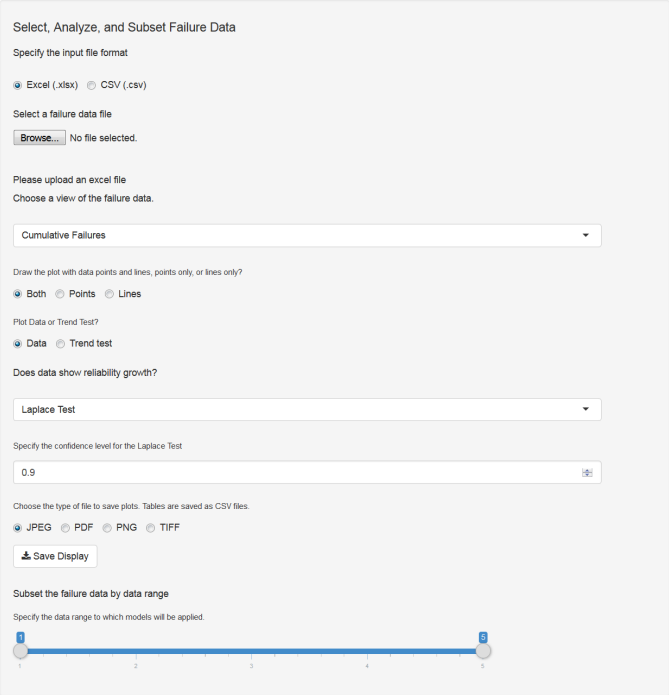


Fig. 2: Tab one options

	A	B	C
1	FN	IF	FT
2	1	3	3
3	2	30	33
4	3	113	146

Fig. 3: Excel input file format

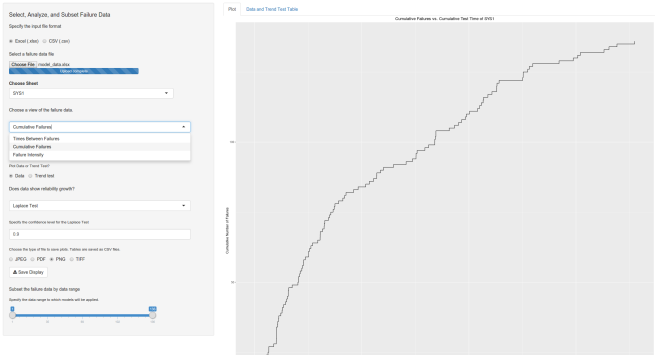


Fig. 4: Tab one after data upload

10 are failure time data and the remaining 21 failure counts. This will enable more comprehensive comparison of models than ever before. Alternative data views include times between failures and failure intensity, which can be selected from the dropdown menu below the prompt to "Choose a view of the failure data". All plots can be drawn using either points, lines or both.

The radio buttons **Data** and **Trend test** allow the user to switch between data and trend test plots. These trend tests have been placed before model fitting to ensure that the data exhibits reliability growth and it is therefore appropriate to apply software reliability models and make predictions. The two tests implemented include the Laplace trend test [8] and running arithmetic average.

Figure 5 shows the Laplace trend test of the SYS1 data set. The red line is a user specified input into the textbox below the prompt to "Specify the confidence level for the Laplace Test" as shown in Figure 2. Here the value has been set to 0.9 or 90%. Additional default levels in black include 90, 95, 99, 99.9, 99.9999, and 99.999999. Values below these lines indicate that the data exhibits reliability growth with the specified level of statistical significance and it is therefore suitable to apply software reliability models.

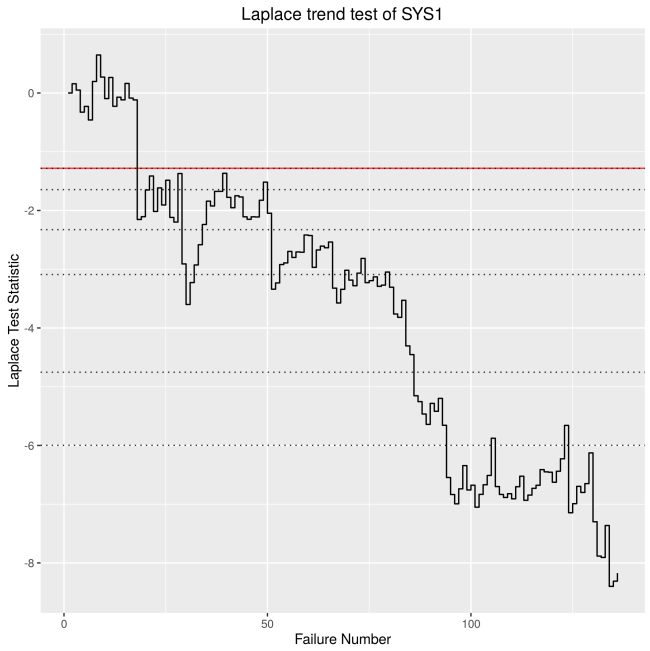


Fig. 5: Laplace trend test

Figure 6 shows the running arithmetic average of the SYS1 data set. Intuitively, if the time between failures increases then the running arithmetic average increases, indicating system reliability is improving. A decreasing running arithmetic average indicates reliability deterioration. Both the Laplace trend test and running arithmetic average suggest the SYS1 data set exhibits reliability growth.

Figure 7 shows the Laplace trend test of the J4 data set which does not experience statistically significant reliability improvement. Thus, it is not appropriate to apply software reliability models to this data set until additional testing is performed that establishes confidence in reliability growth. The slider at the bottom of Figure 2 ranges from 1 to n, where n denotes the number of data points contained in a data set. The sliders allow the user to specify a subset of the data for plotting, model fitting, and prediction. The default is to use all n data points for model fitting.

The user can switch between the **Plot** and **Data and Trend Test Table** as inidcated in Figure 4. Selecting the table view

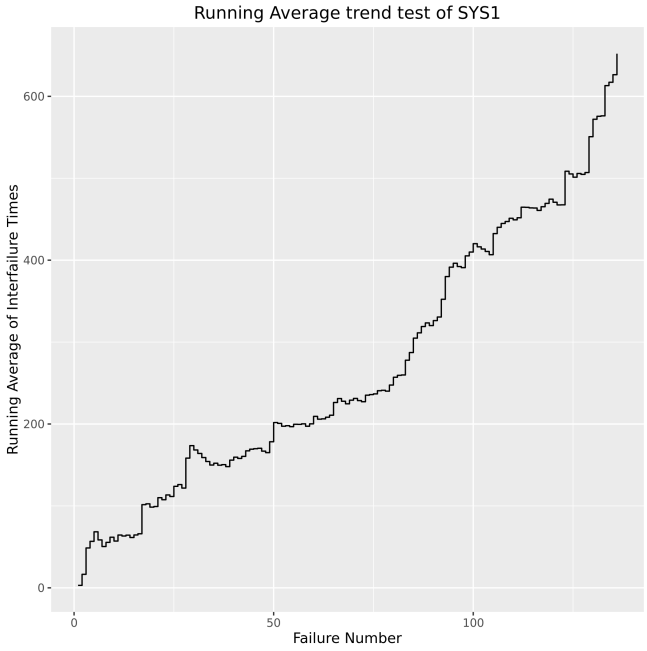


Fig. 6: Running arithmetic average

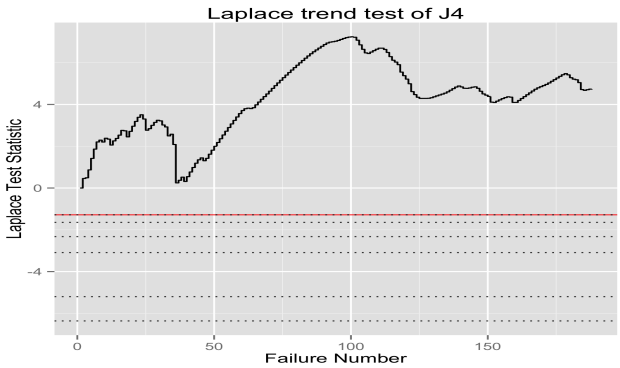


Fig. 7: Data set without reliability growth

displays the numerical data used to draw the plots in a tabular format similar to Figure 3. Selecting a radio button **.jpeg**, **.pdf**, **.png**, and **.tiff** (Figure 2) and then clicking **Save Display** saves a plot in the desired image format, while tables can be saved as csv or pdf. Exporting tabular data enables plotting in third party graphing software for inclusion of images in reports and research papers.

B. Tab 2: Set up and Apply Models

Figure 8 shows the options available on the second tab where model fitting is performed. The first text box allows the user to "Specify the number of failures for which the models will make predictions". Here the number of failures has been set to one for the sake of illustration. The second multi-select box allows the user to identify which models to fit with the data range chosen on Tab one. By default, the tool displays all available models. Clicking the **Run Selected Models** button executes the algorithms to fit selected model.

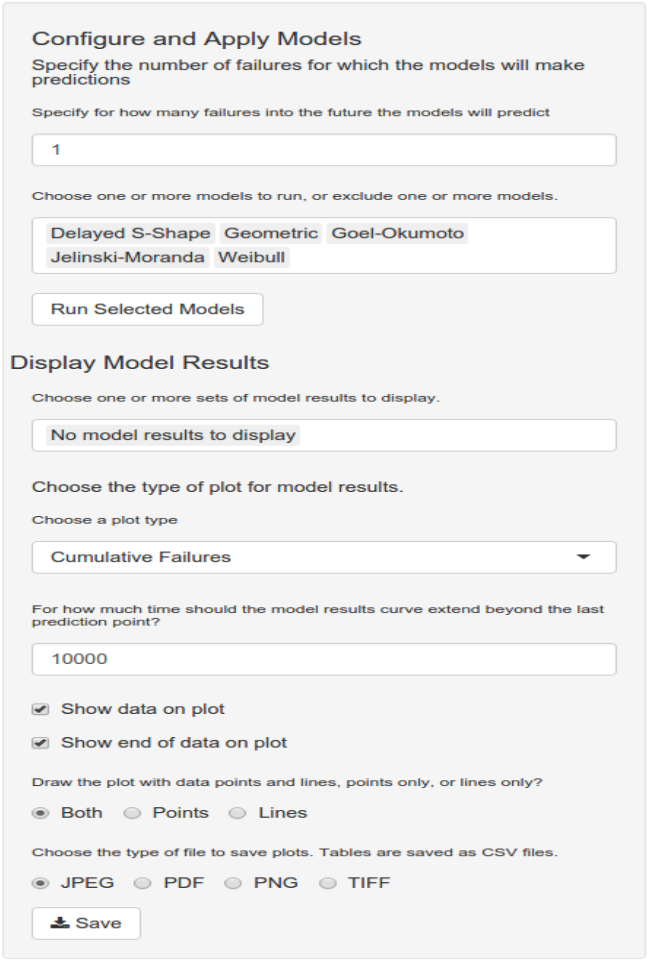


Fig. 8: Tab two options

Once model fitting completes, the multi-select box below "Choose one or more sets of model results to display" is populated with models that completed successfully. Models that fail, however, will not be available. The results of successful models can be compared against the empirical data by selecting "Cumulative Failures", "Time between failures", "Failure intensity", or "Reliability growth" from the dropdown menu below "Choose a plot type".

Figure 9 shows a plot of the observed cumulative failure along with the model fits for all five models. The solid black vertical line indicates the time at which the last failure occured. Thus, points to the right indicate predictions. This line can be hidden by deselecting the **Show end of data on plot** checkbox. The legend at the bottom identifies the line that corresponds to each model fit. Figure 9 suggests that the geometric and Weibull models fit the observed data closely, whereas other models over or under predict the number of failures at various stages of testing.

Figures 10, 11, and 12 show the time between failures, failure intensity, and reliability growth data views respectively

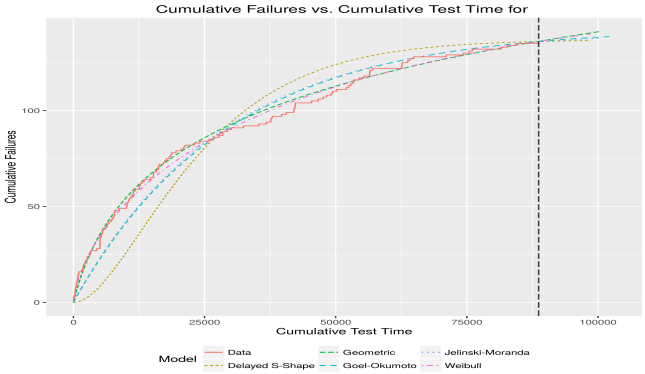


Fig. 9: SYS1 cumulative failures and model fits

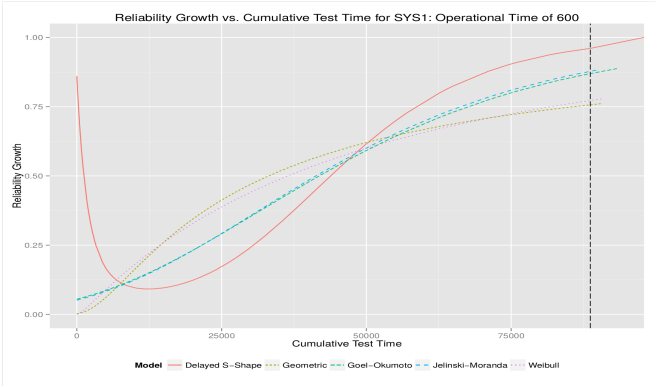


Fig. 12: SYS1 reliability growth and model fits

as well as with the corresponding model fits.

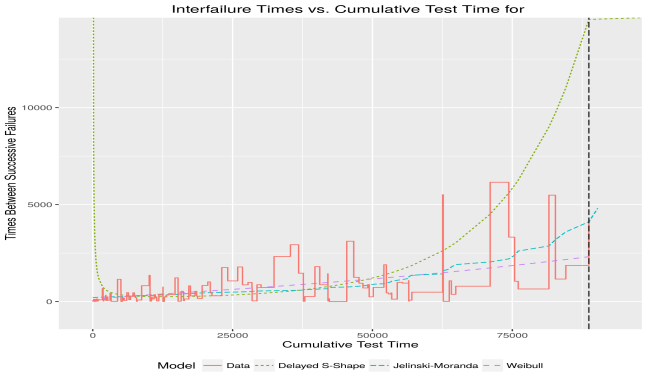


Fig. 10: SYS1 time between failures and model fits

Fig. 11: SYS1 failure intensity and model fits

C. Tab 3: Query Model Results

Figure 13 shows the options on the third tab, which enable a variety of predictions. The multi-select box below "Choose one or more sets of model results to display" allows the user to specify model a with which they would like to make predictions. To determine the time required to observe the next N failures, the user enters the number of failures in the textbox below "Specify the number of failures that are to be observed." Alternatively, the user may "Specify the amount of additional time for which the software will run" to determine the number of faults that would be detected in this interval. Entering numbers into these textboxes automatically generates a table such as Figure 14, which shows the model names and expected number of failures for the next t time units as well as the expected time required to detect the next N failures. In this case, Figure 14 shows predictions for the SYS1 data set, including the time for one additional failure and the predicted number of failures to be observed in the next $100,000$ seconds of additional testing time.

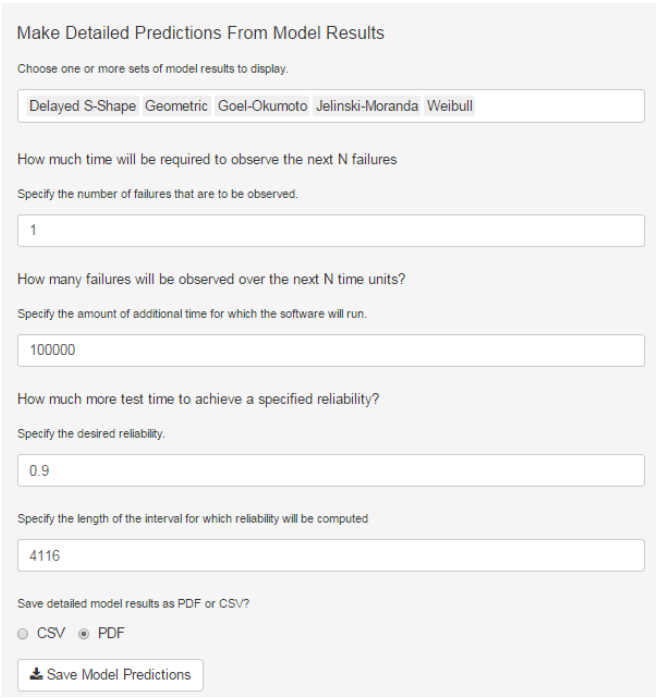
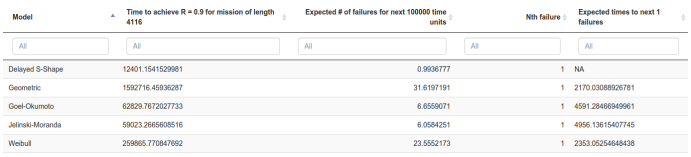


Fig. 13: Tab three options

Tab three (Figure 13) also provides an option to estimate the testing time required to achieve a target reliability by entering the desired reliability and time for which the software must operate without failure in the textbox below "Specify the



Model	Time to achieve R = 0.9 for mission of length 4116	Expected # of failures for next 100000 time units	Nth failure	Expected times to next 1 failures
All	All	All	All	All
Delayed S-Shape	12401.1541529981	0.9936777	1	NA
Geometric	1592716.45936287	31.6197191	1	2170.0308926781
Goel-Okumoto	62829.7672027733	6.6559071	1	4591.28466949961
Jelinski-Moranda	59023.2665608516	6.0584251	1	4956.13615407745
Weibull	259865.770847692	23.5552173	1	2353.05254648438

Fig. 14: Failure predictions

desired reliability" and "How much more test time to achieve a specified reliability?" respectively.

D. Tab 4: Evaluate Models

Figure 15 shows tab four's options, which provide method to assess the relative goodness of fit of alternative models. The multi-select box below **Choose one or more sets of model results** allows the user to specify models to which they would like to apply goodness of fit measures, including the Akaike information criterion (AIC) [9] and predictive sum of squares error (PSSE) [10]. The text box below "Specify the Percent Data for PSSE" allows the user to specify the fraction of the data to be used to compute the PSSE.

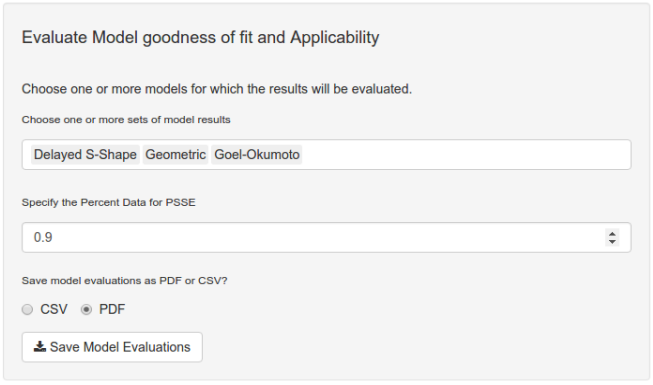


Fig. 15: Tab four options

Figure 16 shows the AIC and PSSE values for all five models when applied to SYS1 data set when 90% percent of data is used to fit models. The up/down arrows next to the performance measures in Figure 16 sort the table based on rankings. Figure 16 indicate that the Geometric and Weibull models perform best with respect to the AIC, while the Jelinski-Moranda and Goel-Okumoto models perform well with respect to PSSE. Like other tabs, the user can save the result table to csv or pdf format.

	Model	AIC	PSSE
	All	All	All
1	Delayed S-Shape	2075.146	296.34925
2	Geometric	1937.034	84.32708
3	Goel-Okumoto	1953.613	23.07129
4	Jelinski-Moranda	1950.541	24.39945
5	Weibull	1938.161	74.94496

Fig. 16: AIC and PSSE of all models

III. Conclusion and Future Research

This paper presents an open source application to promote collaboration among members of the international software reliability research community and users from industry and government organizations. The application architecture will enable incorporation of software reliability models from the research literature into a single tool. Interfailure time, failure time, and failure count data formats are automatically supported. Two trend tests for reliability growth, two failure rate and three failure counting models have been implemented as well as two measures of goodness of fit. Inferences such as the time to achieve a target reliability or detect a specified number of additional failures have also been implemented.

Future research will expand the architecture to enable models for other stages of the software development lifecycle while preserving an intuitive work flow.

Acknowledgment

This work was supported by (i) the Naval Air Systems Command through the Systems Engineering Research Center, a Department of Defense University Affiliated Research Center under Research Task 139 : Software Reliability Modeling and (ii) the National Science Foundation (#1526128).

References

[1] M. Lyu, Ed., *Handbook of Software Reliability Engineering*. New York, NY: McGraw-Hill, 1996.

[2] W. Farr and O. Smith, "Statistical modeling and estimation of reliability functions for software (SMERFS) users guide," Naval Surface Warfare Center, Dahlgren, VA, Tech. Rep. NAVSWC TR-84-373, Rev. 2, 1984.

[3] M. Lyu and A. Nikora, "CASRE: A computer-aided software reliability estimation tool," in *IEEE International Workshop on Computer-Aided Software Engineering*, 1992, pp. 264–275.

[4] H. Okamura and T. Dohi, "SRATS: Software reliability assessment tool on spreadsheet (experience report)," in *International Symposium on Software Reliability Engineering*, Nov 2013, pp. 100–107.

[5] A. Goel, "Software reliability models: Assumptions, limitations, and applicability," *IEEE Transactions on Software Engineering*, no. 12, pp. 1411–1423, 1985.

[6] S. Yamada, H. Ohtera, and H. Narihisa, "Software reliability growth models with testing-effort," *IEEE Transactions on Reliability*, vol. R-35, no. 1, pp. 19–23, apr 1986.

[7] S. Yamada and S. Osaki, "Reliability growth models for hardware and software systems based on nonhomogeneous poisson process: A survey," *Microelectronics and Reliability*, vol. 23, no. 1, pp. 91–112, 1983.

[8] O. Gaudoin, "Optimal properties of the laplace trend test for soft-reliability models," *IEEE Transactions on Reliability*, vol. 41, no. 4, pp. 525–532, 1992.

[9] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[10] L. Fiondella and S. S. Gokhale, "Software reliability model with bathtub-shaped fault detection rate," in *Annual Reliability and Maintainability Symposium*, 2011, pp. 1–6.