Practical Software Reliability Modeling and Application: A Case Study

Vidhyashree Nagaraju, Shekar, Niti, and Lance Fiondella
Department of Electrical and Computer Engineering
University of Massachusetts, Dartmouth, MA, 02747 USA
Email: {vnagaraju,lfiondella}@umassd.edu

Ying Shi
Goddard Space Flight Center
National Aeronautics and Space Administration
Email: @

Abstract—**This paper demonstrates the utility of a script to automatically apply a free and open source software failure and reliability assessment tool. The script generates a pdf report, eliminating the need to work with the user interface of the tool to manually assess the data and report. The reports can be used to summarize progress to both technical and non-technical leadership. Simplifying the assessment and reporting may encourage the software and reliability practitioners to apply the models and make decisions about the process.**

Index Terms—**Software reliability, software reliability growth model, R statistical programming language, GitHub, Software Failure and Reliability Assessment Tool, open source**

I. Introduction

Software reliability growth models (SRGM) [1] are extremely useful in making decisions about software development by characterizing failure data collected during testing. Some of the predictions featured by SRGM include the remaining number of failures, failure intensities, mean time to failures, release time, as well as overall software reliability. However, practitioners may be reluctant to apply these models due to a lack of either the knowledge of the underlying mathematics or the time to develop expertise and implement models in their work. Therefore, several computer-aided software reliability tools [2]–[4] have been developed.

Previous computer-aided tools to apply software reliability methods include: Emerald [5], SRMP (Software Reliability Modeling Programs) [6], the AT&T SRE Toolkit [7], SoRel [8], SMERFS (Statistical Modeling and Estimation of Reliability Functions for Software) [2], and CASRE (Computer Aided Software Reliability Estimation) [4], Robust [9], SREPT [10], CARATS [11], SRATS [3], and M-SRAT [12]. Most of these tools are mostly spreadsheet based [3] or close source in nature. This inhibits the capability to integrate the existing tools into software testing work flows and update the tools with recent changes in the software reliability research. To overcome the limitation of existing tools, an open source Software Failure and Reliability Assessment Tool (SFRAT) [13] is developed with a flexible architecture to accommodate individual researchers models as well as methods.

In this paper, we present a script to automatically apply an open source SFRAT and generate report. This script eliminates the need to work with the graphical user interface to manually prepare the report, thus conserving time. The script can be configured to produce custom reports; for example, the user can opt to include explanations of each result in the report. The script generates a pdf to visually assess the data and to ease the reporting process. The reports can then be used to summarize and visualize project status to both technical and non-technical leadership.

This work is supported by NASA under contract number:

The remainder of the paper is organized as follows: Section II provides a brief overview of the SFRAT user interface and Section III provides a detailed discussion of the script to generate the report automatically. Section IV demonstrates the use of the script through a real data, while Section V provides conclusion and directions for future research.

II. Software Failure and Reliability Assessment Tool (SFRAT)

The Software Failure and Reliability Assessment Tool is a free and open source tool developed to promote quantitative assessment of software reliability as well as improved communications of such assessments. SFRAT is an application that estimates and predicts the reliability of a software system during test and operation. Some of the questions that the tool answers about a system undergoing test are:

- Is the software reliable enough to release?
- How long will it be before a specified goal is reached?
- What will be the consequences if testing resources are insufficient?

SFRAT is implemented in the R statistical programming language [], an open source environment for statistical computing and graphics that can be used on computers running Windows, OSX, or Linux, with the user interface implemented using the R Shiny library []. The code is accessible on GitHub at *https://github.com/LanceFiondella/srt.core*, which after downloading may run using RStudio or some other R environment. In doing so, an organization can easily perform information assurance of the code prior to use on sensitive failure data. For complete details, the reader is referred to *http://sasdlc.org/lab/projects/srt.html*. The architecture of the SFRAT combines the use of existing software reliability models into a single tool, enabling more systematic comparison of models than previously possible. Presently, data formats supported consist of inter failure time, failure time, and failure count. Functionality includes two trend tests for reliability growth, two failure rate [1], Jelinski-Moranda and geometric, and three failure counting models, Goel-Okumoto [14], delayed S-shaped [15], and Weibull [16] models as well as two measures of goodness of fit. Also implemented are methods to predict time to reach a desired reliability as well as detecting a number of future failures. The free and open source nature of the tool architecture enables additional models and goodness of fit measures to be later implemented. In this regard, the tool is intended to serve as a shared environment for collaboration among researchers and practitioners.

SFRAT can be accessed by downloading the source code from GitHub and run using an R environment (e.g. RStudio), or through the web instance. Once the application is launched, the initial SFRAT screen is shown as seen in Figure 1.

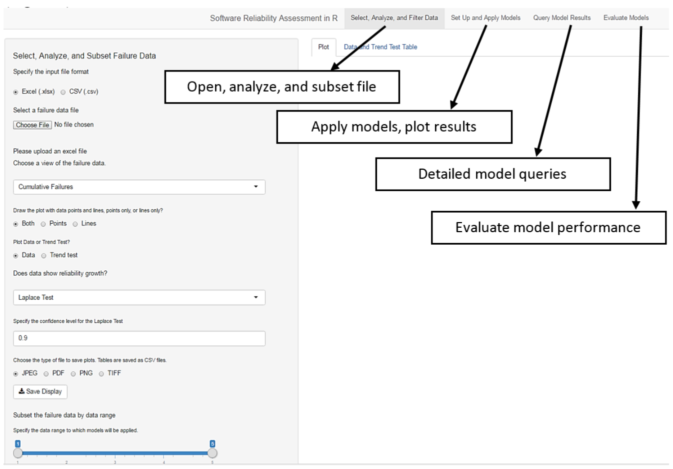

Software Reliability Assessment in R
Select, Analyze, and Filter Data
Set Up and Apply Models
Query Model Results
Evaluate Models
Plot
Data and Trend Test Table
Open, analyze, and subset file
Apply models, plot results
Detailed model queries
Evaluate model performance

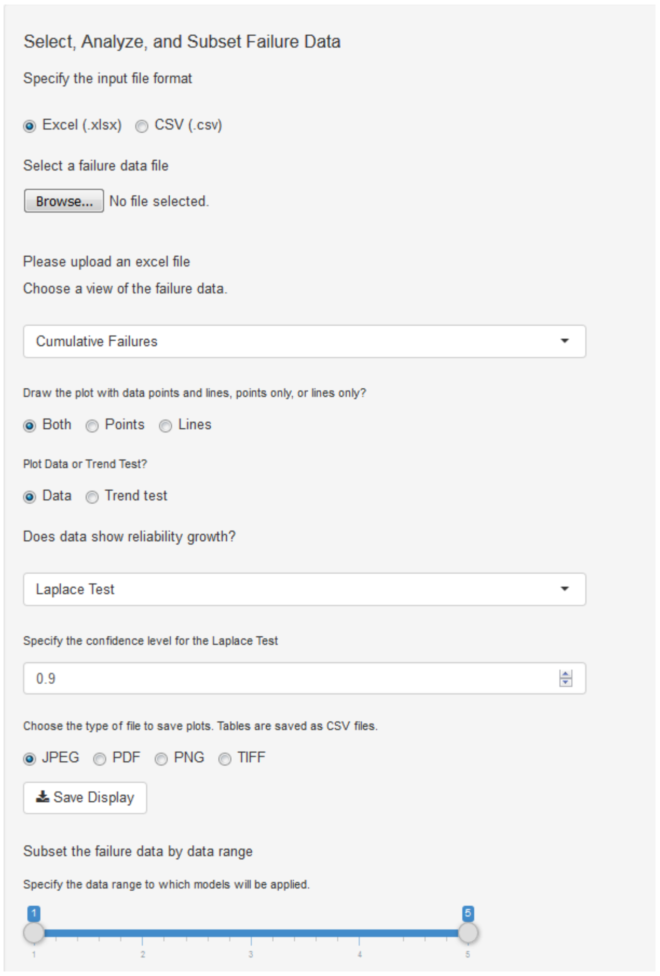
Fig. 1: Initial View within SFRAT

The SFRAT workflow is divided into four subtasks:

- **Select, Analyze, and Filter Data**: Import failure data history and determine model applicability of its reliability's growth.
- **Set Up and Apply Models**: View results of one or more reliability models, including reliability growth.
- **Query Model Results**: Query applied models to answer the following questions:
  - How many additional failures will occur during a continued period of testing?
  - How much time would be required to observe a number of additional failures?
  - At what point in testing will a reliability goal be reached?
- **Evaluate Models**: By model comparison, determine which is most likely to proficiently characterize the data and provide accurate predictions.

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

Figure 2 shows the options in the first tab, which includes file selection, data visualization, and trend test analysis. The user can then 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 failure number(FN), interfailure time(IF), and failure time(FT) are indicated.


Select, Analyze, and Subset Failure Data

Specify the input file format

[✓ Excel (.xlsx)] CSV (.csv)

Select a failure data file

Browse... No file selected.

Please upload an excel file

Choose a view of the failure data.

Cumulative Failures

Draw the plot with data points and lines, points only, or lines only?

[✓ Both] Points Lines

Plot Data or Trend Test?

[✓ Data] Trend test

Does data show reliability growth?

Laplace Test

Specify the confidence level for the Laplace Test

0.9

Choose the type of file to save plots. Tables are saved as CSV files.

[✓ JPEG] PDF PNG TIFF

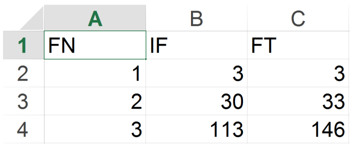
Save Display

Subset the failure data by data range

Specify the data range to which models will be applied.

1 — 5

Fig. 2: Tab one view

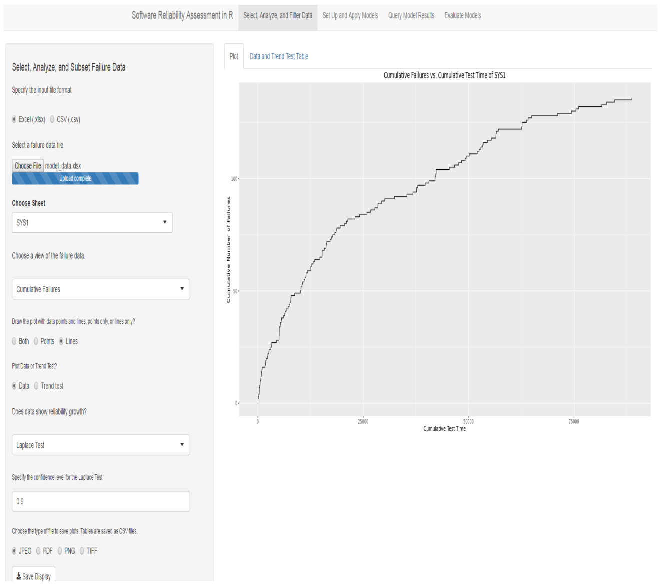


	A	B	C
1	FN	IF	FT
2	1	3	3
3	2	30	33
4	3	113	146

Fig. 3: Example Excel input

Clicking on the **Browse...** button enables the user to search for and upload the input data file. The progress bar message "Upload complete" will indicate a successful file upload. By default, a plot of the cumulative failures for the uploaded data set is displayed. For example, Figure 4 shows the SYS1 data set [1].

From a drop-down menu below the progress bar, the user may choose from the provided data sets. CSV files may only contain only one data set, whereas Excel files will contain one data set per sheet. Data sets that do not comply with the input format will not be available in the dropdown menu. Regardless of the input's data format, the tool will provide


Software Reliability Assessment in R
Select, Analyze, and Filter Data
Set Up and Apply Models
Query Model Results
Evaluate Models
Plot
Data and Trend Test Table
Cumulative Failures vs. Cumulative Test Time of SYS1
Cumulative Number of Failures
Cumulative Test Time
0
50
100
0
25000
50000
75000

Select, Analyze, and Subset Failure Data

Specify the input file format

[✓ Excel (.xlsx)] CSV (.csv)

Select a failure data file

Choose File model_data.xlsx

Upload complete

Choose Sheet

SYS1

Choose a view of the failure data.

Cumulative Failures

Draw the plot with data points and lines, points only, or lines only?

Both Points [✓ Lines]

Plot Data or Trend Test?

[✓ Data] Trend test

Does data show reliability growth?

Laplace Test

Specify the confidence level for the Laplace Test

0.9

Choose the type of file to save plots. Tables are saved as CSV files.

[✓ JPEG] PDF PNG TIFF

Save Display

Fig. 4: Tab one view after upload

conversions to failure time, failure count, and inter failure data formats upon upload, allowing for the use of any data model. 31 data sets were taken from the Handbook of Software Reliability Engineering [1] and prepared in the file format. Of these, 10 are failure time data and the remaining 21 are failure counts. This enables a more comprehensive comparison between models than previously possible. Using the drop-down menu below "Choose a view of the failure data", the user may select an alternative data view, including failure intensities and the times between failures. In addition, the option exists to draw plots with either points, lines or both.

Data and Trend test plotting is also possible by use of the radio buttons with the same labels. These trend tests have been placed before model fitting to ensure that the data exhibits reliability growth and is therefore appropriate for software reliability models and predictions. The two tests implemented include the Laplace trend test [17] as well as a running arithmetic average.

Figure 5 shows the Laplace trend test of the SYS1 data set. The red line seen is a user-specified input, entered into the text box just underneath the Laplace Test drop-down box, as shown in Figure 2. In this case, 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 is therefore suitable to apply software reliability models.

Figure 6 shows the running arithmetic average of the SYS1 data set. If the time between failures increases, the running arithmetic average then increases, indicating an increase in system reliability. A decreasing running arithmetic average indicates reliability deterioration. Both the Laplace trend test and running arithmetic average suggest that the example SYS1 data set exhibits reliability growth.

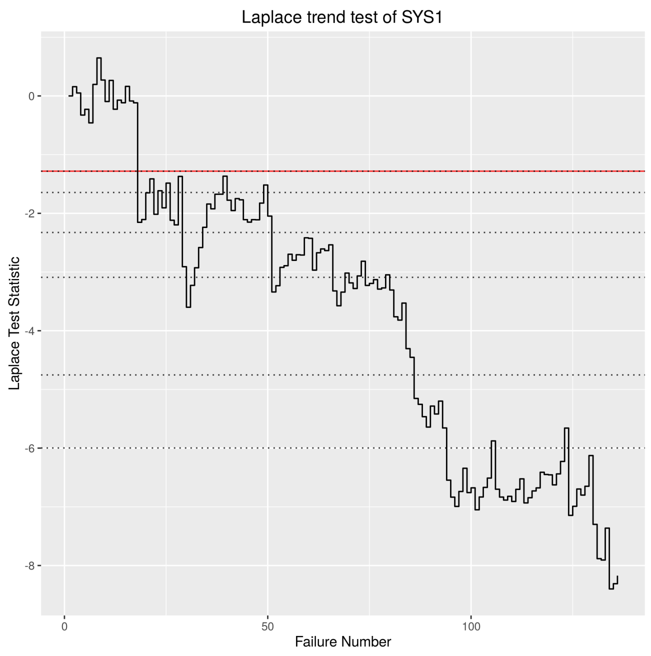


Fig. 5: Laplace trend test

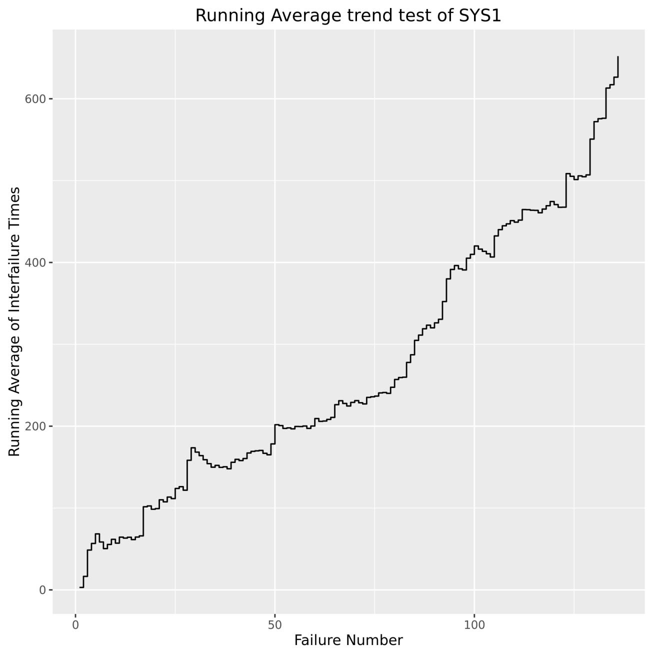


Fig. 6: Running arithmetic average

Figure 7 shows the Laplace trend test of the J4 data set, which does not experience any significant improvement in reliability. Thus, it is not appropriate to apply software reliability models to this data set until additional testing is performed to establish 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.

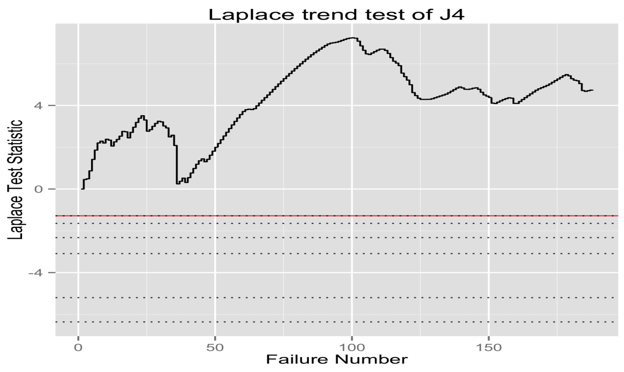


Fig. 7: Data set without reliability growth

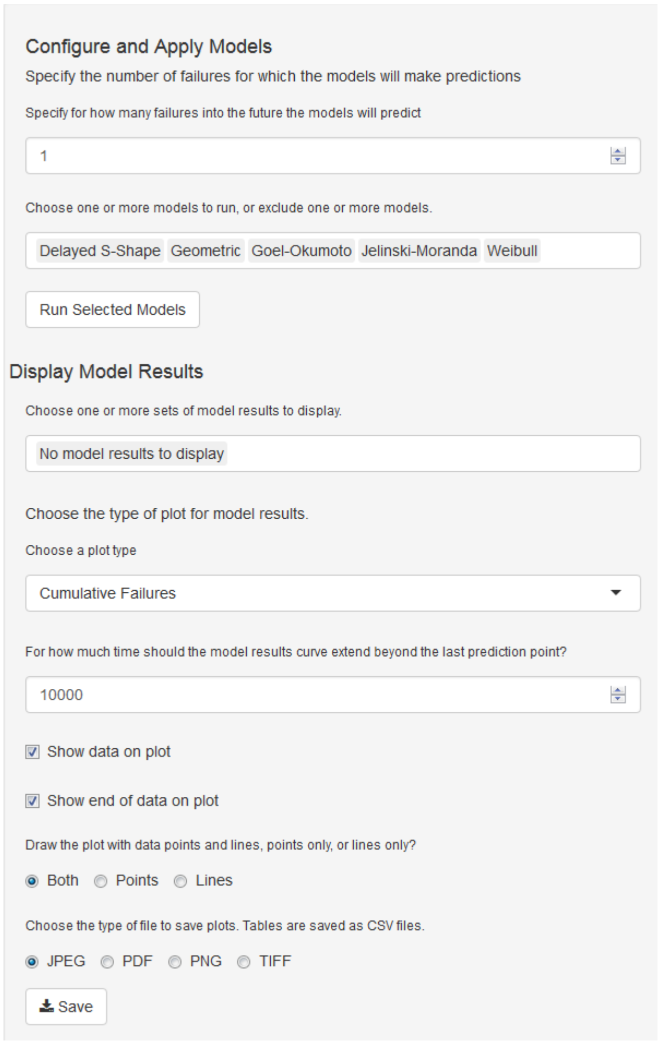
The user can switch between the **Plot** and **Data and Trend Test Table** as shown in Figure 4. Selecting table view displays the raw numerical data used to draw the plots in a tabular format similar to Figure 3. Selecting a radio button **.jpeg**, **.pdf**, **.png**, or **.tiff** (Figure 2) and then clicking **Save Display** saves a plot in the selected image format, while numerical tables may be saved as a CSV or PDF file. Tabular data may be exported to input into graphing software to include images in reports and research papers.

B. Tab 2: Set up and Apply Models

Figure 8 shows the second tab's options, where model fitting will be 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 the selected model.

Once model fitting completes, the multi-select box below "Choose one or more sets of model results to display" is populated with models that have 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 failures along with the fitted models. The black vertical line indicates the time at which the last failure occurred. Thus, points to its right indicate predicted failures. 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-predict or under-predict the number of failures at various stages of testing.



Configure and Apply Models

Specify the number of failures for which the models will make predictions

Specify for how many failures into the future the models will predict

1

Choose one or more models to run, or exclude one or more models.

Delayed S-Shape Geometric Goel-Okumoto Jelinski-Moranda Weibull

Display Model Results

Choose one or more sets of model results to display.

No model results to display

Choose the type of plot for model results.

Choose a plot type

Cumulative Failures

For how much time should the model results curve extend beyond the last prediction point?

10000

[✓] Show data on plot

[✓] Show end of data on plot

Draw the plot with data points and lines, points only, or lines only?

[✓ Both] Points Lines

Choose the type of file to save plots. Tables are saved as CSV files.

[✓ JPEG] PDF PNG TIFF

Fig. 8: Tab two options

Figures 10, 11, and 12 show the times between failures, failure intensities, and reliability growth data views respectively as well as with the corresponding model fits.

C. Tab 3: Query Model Results

Figure 13 shows the options on the third tab, which focus on a variety of predictive algorithms. The multi-select box below "Choose one or more sets of model results to display" allows the user to specify a model with which they would like to make predictions. To determine the time required to observe the next N failures, the user enters that number of failures in the text box just below. 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 a specified interval. Entering numbers into these text boxes automatically generates a table (for example see 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

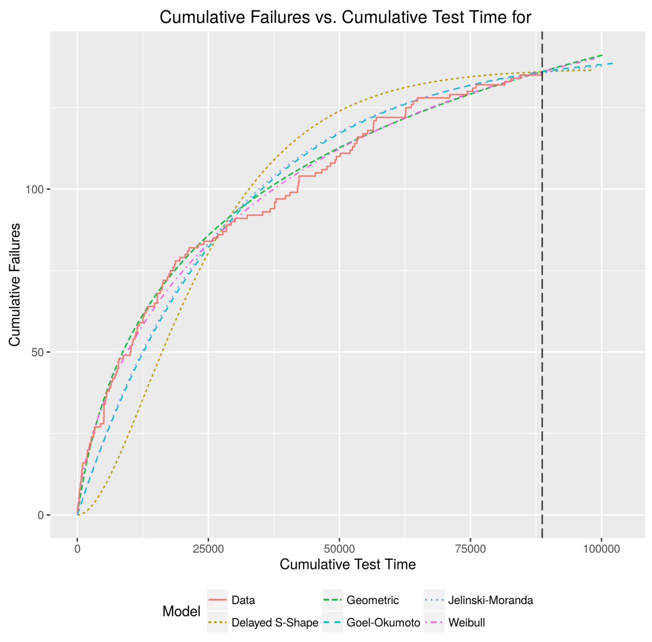


Fig. 9: SYS1 cumulative failures and model fits

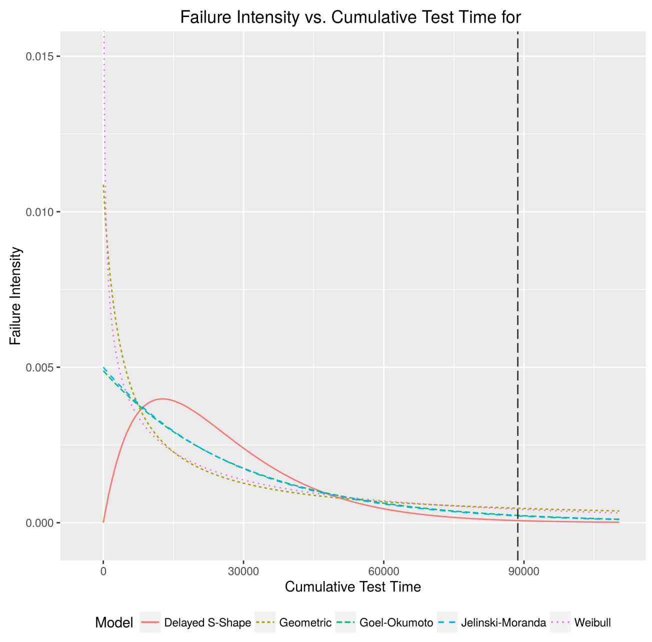


Fig. 11: SYS1 failure intensity and model fits

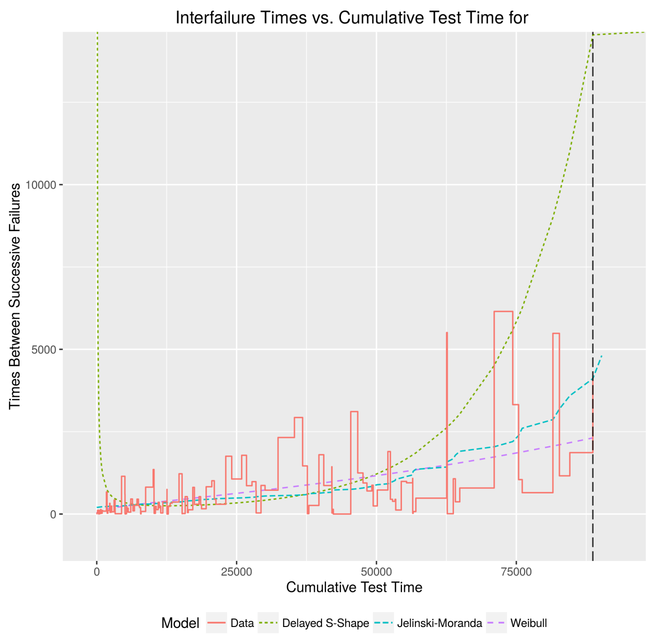


Fig. 10: SYS1 time between failures and model fits

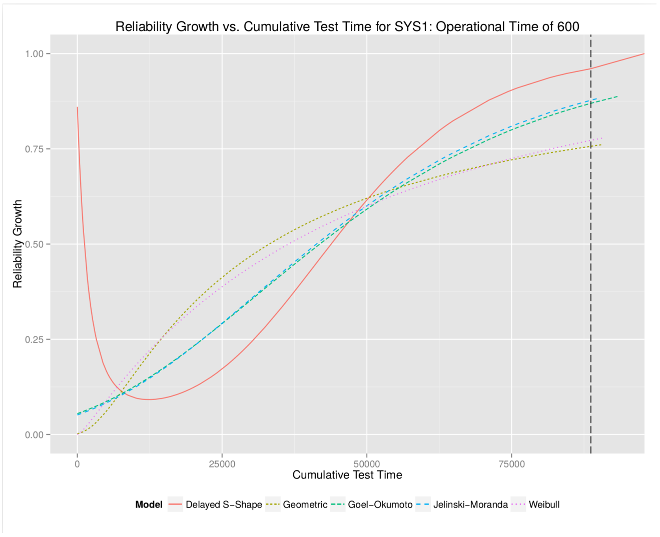


Fig. 12: SYS1 reliability growth and model fits

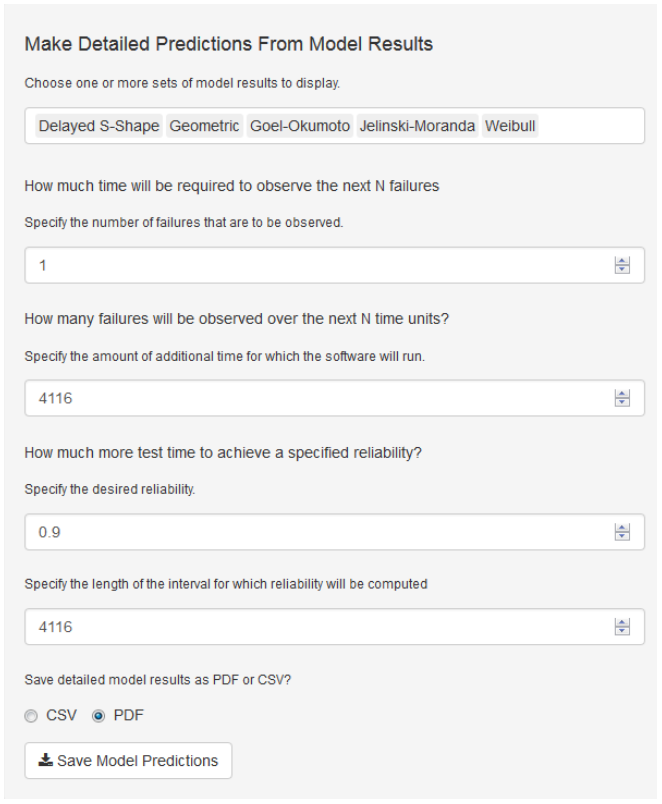
additional failure and the predicted number of failures to be observed in the next $100,000$ seconds of additional testing time.

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 text box below “Specify the 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 methods to assess model goodness of fit. The multi-select box below **Choose one or more sets of model results** allows a user to specify models they would like to apply goodness of fit measures, including the Akaike information criterion (AIC) [18] and predictive sum of squares error (PSSE) [19]. The text box below “Specify the Percent Data for PSSE” allows specification of the fraction of data to be used to compute PSSE.

Figure 16 shows the AIC and PSSE values for the five models applied to SYS1 when 90% of data is used. The



Make Detailed Predictions From Model Results

Choose one or more sets of model results to display.

Delayed S-Shape, Geometric, Goel-Okumoto, Jelinski-Moranda, Weibull

How much time will be required to observe the next N failures

Specify the number of failures that are to be observed.

1

How many failures will be observed over the next N time units?

Specify the amount of additional time for which the software will run.

4116

How much more test time to achieve a specified reliability?

Specify the desired reliability.

0.9

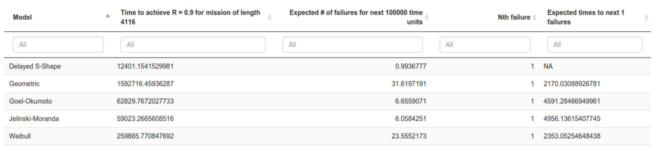
Specify the length of the interval for which reliability will be computed

4116

Save detailed model results as PDF or CSV?

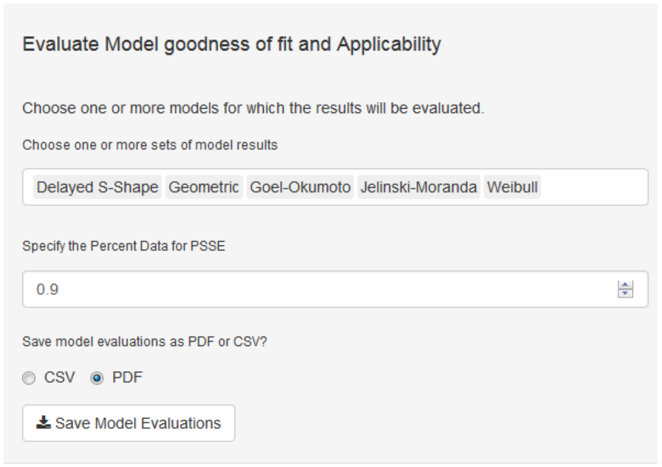
CSV [✓ PDF]

Fig. 13: Tab three options



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	2170.03088926781
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



Evaluate Model goodness of fit and Applicability

Choose one or more models for which the results will be evaluated.

Choose one or more sets of model results

Delayed S-Shape, Geometric, Goel-Okumoto, Jelinski-Moranda, Weibull

Specify the Percent Data for PSSE

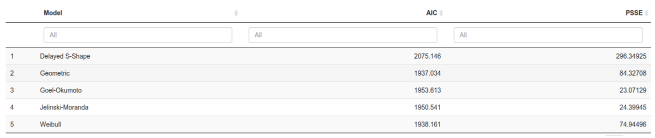
0.9

Save model evaluations as PDF or CSV?

CSV [✓ PDF]

Fig. 15: Tab four options

up/down arrows next to the performance measures in Figure 16 sort the table based on rankings. Figure 16 indicates 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.



	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. SFRAT REPORT GENERATION SCRIPT

This section discusses the usage of the automated SFRAT script. This script will run the SFRAT based on user-defined variables inside of the script, and will then generate a PDF report based on the SFRAT's output. Manually configuring the settings via the UI and then creating a report based on the tool's output may be tedious and unpleasant, so users may wish for something quicker and easier. Using the script, it becomes simple to repeatedly generate reader-friendly reports of reliability growth data. The report generation script is written in the R programming language, and uses the Markdown library to graphically generate the report. To achieve this, two files are used, being **report-specifications.R** and **SFRATReport.Rmd**. The **report-specifications.R** is used to configure the variables taken into account when generating the report, and the **SFRATReport.Rmd** Markdown file is used as a template when placing the tool's output into a readable PDF report.

A. Using the report generation script

To run the main script using RStudio, launch RStudio and set its working directory to the location where the folder is saved via **Session** → **Set Working Directory**. Ensure that the required packages are installed. See the Documentation for the list of packages. In the file **report-specifications.R**, each variable represents some parameter in the UI for the SFRAT. Any setting found in the UI will be configurable inside this script. Below is a brief description of each parameter found in the script. Main script parameters:

- **verbose_report** - Enabling this setting will show verbose reports on the PDF report. It provides a brief description of the purpose of each figure.
- **datapath** - See "Select a failure data file" in Figure 2 - this denotes the location of the failure data file.
- **sheetNumber** - This allows the user to choose which data set is analyzed, given that the supplied file has more than one. This is the equivalent of "Choose Sheet" in the SFRAT's first tab.
- **colors** - This specifies the set of colors used in the report.

Tab 1 options:

- **confidence_lvl** - This allows the user to specify a confidence level between 0 and 1 for the Laplace trend test to quantify a desired level of significance for reliability growth. See Figure 5 for an example.

Tab 2 options:

- **num_failures_future_prediction** - The user can specify the number of failures that they would like to predict beyond the end of testing. This would be the top option seen in Figure 8.
- **models_to_apply** - This allows the user to specify which software reliability growth models they would like to apply to make predictions. The available models are a delayed s-shape (DSS), geometric (GM), Goel-Okumoto (GO), Jelinski-Moranda (JM), and Weibull (Wei) model.
- **mission_time** - This will extend the plot axis this amount of time past the data ending to be able to view the reliability growth predictions. See towards the right of vertical dotted line in Figure 9 for example.

Tab 3 options:

- **num_failures_to_predict** - Specify the number of failures to predict beyond the end of testing. This is similar to *num_failures_future_prediction*.
- **additional_time_software_will_run** - User can specify the additional time beyond the end of testing to predict the number of failures.
- **desired_reliability** - User can specify the target reliability between 0 to 1 to estimate the time required to achieve such failures.
- **reliability_interval_length** - This is used to specify the mission time beyond testing to estimate the reliability. See Figure 13 for examples.

Tab 4 options:

- **percent_data_for_PSSE** - The user can specify the percentage of data to be used for model fitting. The remaining data will be used to assess model prediction capability using the predictive sum of squares error.

To modify the header of the report PDF (e.g. author name), modify lines 2 through 6 in **SFRATReport.Rmd**. If the user so wishes, other parameters within the Rmd file can be changed to modify the report. After saving the input specification script, open it in RStudio and click "Source" (top-right of the script editor).

The generated reports will be stored in the *Report* folder, which is located in the same directory as the script. Each report file will have the following naming convention:
SFRAT report_dataName_YYYY-MM-DD.pdf

The script may also be easily run via the command line. Assuming that all prerequisites are met, navigate to the script folder, run R and enter *source('report-specifications.R')*.

IV. Illustrations

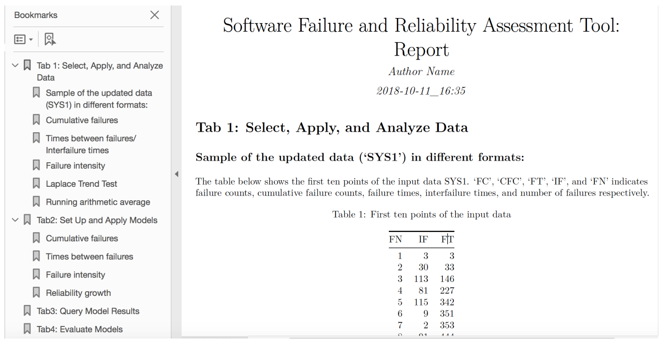
This section displays the example results based on the SYS1 example data [1]. The second example demonstrates the online assessment of the data using SYS1.

A. Script on SYS1 data

This example demonstrates the use of the script in the context of SYS1 example data, which consists of 136 failures. The in-line comments have been removed as variable summaries are available in Section III-A. This example purely displays the example values used to generate the given report. The following variables used work best with the SYS1 data set.

```
verbose_report <- TRUE
sheetNumber <- 1
filePath <-
    '/SFRAT/model_testing/model_data.xlsx'
colors <-
    c("navy","red","green","firebrick4","magenta")
confidence_lvl <- 0.9
num_failures_future_prediction <- 2
models_to_apply <- c('DSS', 'GM',
    'Wei','GO','JM')
mission_time <- 600
num_failures_to_predict <- 2
additional_time_software_will_run <- 4116
desired_reliability <- .9
reliability_interval_length<- 600
percent_data_for_PSSE <- .9
```

Figure 17 shows the initial view of the SFRAT report generated, displaying an initial view of some of the input SYS1 data. This provides a verification as well as a sample of the data used.


Bookmarks
Tab 1: Select, Apply, and Analyze Data
Sample of the updated data (SYS1) in different formats:
Cumulative failures
Times between failures/ Interfailure times
Failure intensity
Laplace Trend Test
Running arithmetic average
Tab2: Set Up and Apply Models
Cumulative failures
Times between failures
Failure intensity
Reliability growth
Tab3: Query Model Results
Tab4: Evaluate Models

Software Failure and Reliability Assessment Tool: Report

Author Name

2018-10-11_16:35

Tab 1: Select, Apply, and Analyze Data

Sample of the updated data ('SYS1') in different formats:

The table below shows the first ten points of the input data SYS1. 'FC', 'CFC', 'FT', 'IF', and 'FN' indicates failure counts, cumulative failure counts, failure times, interfailure times, and number of failures respectively.

Table 1: First ten points of the input data

FN	IF	FT
1	3	3
2	30	33
3	113	146
4	81	227
5	115	342
6	9	351
7	2	353

Fig. 17: Initial view of the report using SYS1 data

When viewing the PDF, the left side of the report (as demonstrated in Figure 17) will give a outline of the different sections of the report. This particular report is presented over 12 pages, with each result using a separate page. With verbose reporting enabled, a brief description of each graph or table is also presented. See Figure 18 for an example of non-verbose reporting - opposite to what's seen in Figure 17. In addition, the first page allows the user to specify the author, report name, date, and time. If not previously done, modify the first few lines in **SFRATReport.Rmd** inside the script folder and re-generate the report.

To describe the report shortly, graphs for times between failures and failure intensity as a function of the testing time is displayed on pages 3 and 4 respectively. Similar plot and discussion for the running arithmetic average is included in the following page.

V. Conclusion and Future Work

In this paper, we presented a script to automatically apply the Software Failure and Reliability Assessment Tool. An example of the SFRAT in the context of SYS1 dataset and

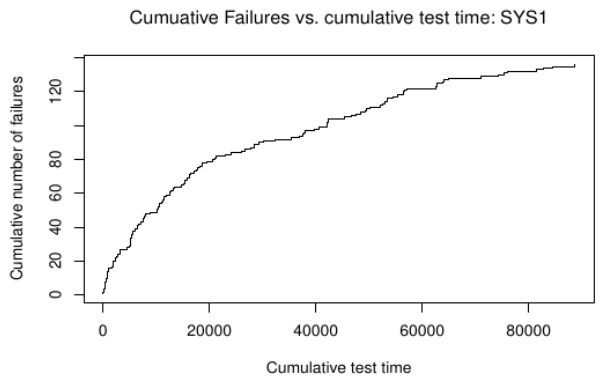


Fig. 18: Example report page with verbose output disabled

a detailed description to run the tool is discussed. Automated PDF report generation is demonstrated using SYS1 data set. A second example is presented to illustrate the utilization of script to perform online assessment of the data.

Future work will extend this script to accommodate the updated version of the SFRAT. Both the SFRAT and the script will be ported to the Python programming language, considering its ease of use. In addition, options to perform online data assessment will be integrated with required functionality and measures.

Documentation

Script user requirements:

- *Operating System*: Windows 7 (64-bit) or newer, Mac OS X 10.9 or newer, Linux capable of running R
- *Perl*: Perl 5 version 16 or newer - may be pre-installed on OSX/Linux; for Windows, Perl may be downloaded from *http://strawberryperl.com/*
- *LaTeX*: Some user-chosen LaTeX editor (e.g. MikTex, TexStudio, Texmaker)
- *R*: R(ver. 3.0 or newer) and RStudio(ver. 0.99.482 or newer) are required, downloadable from *http://rstudio.com*.
- *pandoc*: Newer versions of R do not support installing pandoc as a package, so it must be downloaded separately from *https://pandoc.org/installing.html*.

In addition, some R packages must be installed before script use. Either run **installscript.R** using RStudio (via the *"Source"* option, or command line, *Rscript installscript.R*), or navigate to the "Install packages..." option in the "Tools" menu. To install packages manually, enter the package names into the text input, ensuring that "Install dependencies" is selected. It's unnecessary to change the installation source or location. These packages must be installed before usage:

- **shiny**, a web application framework for R
- **gdata**, a package providing data manipulation tools
- **ggplot**, a graphical package capable of creating elegant and complex plots
- **DT**, a package providing an R interface to the JavaScript library DataTables
- **rootSolve**, a package used to find the roots of n nonlinear (or linear) equations
- **knitr**, a package providing a general-purpose tool for dynamic report generation in R using Literate Programming techniques
- **rmarkdown**, a package allowing converting R Markdown documents into a variety of formats including HTML, MS Word PDF, and Beamer
- **markdown**, a package for authoring HTML, PDF, and MS Word documents
- **readxl**, a package to import excel files
- **formatR**, a package designed to reformat R code to improve readability

Acknowledgment

This work is supported by the National Aeronautics and Space Administration (NASA) and SARP under Grant Number (#). The authors would also like to acknowledge Christian Ellis, Joshua Steakelum, and Melanie Luperon for their inputs on the manuscript.

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] 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.

[4] 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.

[5] J. Hudepohl, S. Aud, T. Khoshgoftaar, E. Allen, and J. Mayrand, "Emerald: Software metrics and models on the desktop," *IEEE Software*, vol. 13, no. 5, pp. 56–60, 1996.

[6] Reliability and S. C. Ltd., "Software reliability modeling programs, Version 1.0," Tech. Rep., 1988.

[7] A. B. Laboratories, "Draft software reliability engineering: Reliability estimation tools reference guide Version 3.7," Tech. Rep., 1990.

[8] K. Kanoun, M. Kaaniche, J. Laprie, and S. Metge, "Sorel: A tool for reliability growth analysis and prediction from statistical failure data," in *IEEE International Symposium on Fault-Tolerant Computing*, 1993, pp. 654–659.

[9] N. Li and Y. Malaiya, "ROBUST: A next generation software reliability engineering tool," in *IEEE International Symposium on Software Reliability Engineering*, 1995, pp. 375–380.

[10] S. Ramani, S. Gokhale, and K. Trivedi, "SREPT: Software reliability estimation and prediction tool," *Performance evaluation*, vol. 39, no. 1-4, pp. 37–60, 2000.

[11] C.-Y. Huang and M. Lyu, "Estimation and analysis of some generalized multiple change-point software reliability models," *IEEE Transactions on reliability*, vol. 60, no. 2, pp. 498–514, 2011.

[12] K. Shibata, K. Rinsaka, and T. Dohi, "M-SRAT: Metrics-based software reliability assessment tool," *International Journal of Performability Engineering*, vol. 11, no. 4, pp. 369–379, 2015.

[13] V. Nagaraju, K. Katipally, R. Muri, T. Wandji, and L. Fiondella, "An open source software reliability tool: A guide for users," in *International Conference on Reliability and Quality in Design*, CA, Aug 2016, pp. 132–137.

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

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.