SoftwareX 10 (2019) 100357

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Practical software reliability engineering with the Software Failure and Reliability Assessment Tool (SFRAT)

Vidhyashree Nagaraju [a], Venkateswaran Shekar [a], Joshua Steakelum [a], Melanie Luperon [a], Ying Shi [b], Lance Fiondella [a,*]

[a] Department of Electrical and Computer Engineering, University of Massachusetts, Dartmouth, MA, 02747, USA
[b] Goddard Space Flight Center, National Aeronautics and Space Administration, USA

ARTICLE INFO

Article history:
Received 20 August 2019
Received in revised form 3 November 2019
Accepted 4 November 2019

Keywords:
Software reliability engineering
Software reliability growth model
Software Failure and Reliability Assessment Tool
Open source
R programming language
GitHub

ABSTRACT

Many large software projects struggle to achieve their reliability targets. Software reliability growth models quantify reliability from failure data collected during software testing. This paper presents the Software Failure and Reliability Assessment Tool (SFRAT), which implements several software reliability growth models as a free and open source application. The open source nature of the tool enables users to integrate the methods into their organization's workflow and researchers to contribute additional statistical methods. The tool is presented in the context of a NASA project.

Code metadata

Current code version	v1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_210
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	R
Compilation requirements, operating environments & dependencies	https://github.com/LanceFiondella/SFRAT-Automated-Report/blob/master/installscript.R
If available Link to developer documentation/manual	https://sasdlc.org/lab/assets/projects/srt.html
Support email for questions	lfiondella@umassd.edu

Software metadata

Current software version	v1.0
Permanent link to executables of this version	https://github.com/LanceFiondella/SFRAT-Automated-Report
Legal Software License	MIT License
Computing platforms/Operating Systems	Linux, OS X, Microsoft Windows, and Unix-like
Installation requirements & dependencies	Refer to user's guide
If available, link to user manual – if formally published include a reference to the publication in the reference list	https://sasdlc.org/lab/assets/projects/srt.html
Support email for questions	lfiondella@umassd.edu

* Corresponding author.
E-mail addresses: vnagaraju@umassd.edu (V. Nagaraju), vshekar@umassd.edu (V. Shekar), jsteakelum@umassd.edu (J. Steakelum), mluperon@umassd.edu (M. Luperon), ying.shi@nasa.gov (Y. Shi), lfiondella@umassd.edu (L. Fiondella).

https://doi.org/10.1016/j.softx.2019.100357

1. Motivation and significance

Many large software projects do not achieve their reliability targets or simply fail. Some industry experts [1] attribute this to poor reporting that leads to suboptimal outcomes, wasted developer time, and dissatisfied customers as well as lawsuits. Methods to assess software reliability can complement design and testing efforts intended to produce quality software.

Software reliability growth modeling [2] is a quantitative approach to characterize failure data collected during testing. Some predictions enabled by Software reliability growth models (SRGM) include number of unique defects remaining, failure intensity, mean time to failure, and software reliability, defined by the American National Standards Institute (ANSI) [3] as the probability of failure-free software operation for a specified period of time in a specified environment. However, software engineers may be reluctant to apply SRGM due to a lack of awareness, knowledge of the underlying mathematics, or time to develop expertise and apply models as they work.

To promote the application of software reliability engineering methods, several computer-aided software reliability tools have been developed, including Emerald [4], SRMP (Software Reliability Modeling Programs) [5], the AT&T SRE Toolkit [6], SoRel [7], SMERFS (Statistical Modeling and Estimation of Reliability Functions for Software) [8], and CASRE (Computer Aided Software Reliability Estimation) [9], Robust [10], SREPT [11], CARATS [12], SRATS [13], and M-SRAT [14]. More recently, Cinque et al. [15] proposed Debugging-Workflow-Aware SRGM, which utilize data from issue tracking systems to improve analytical model predictions as well as support debugging process improvement decisions, while Carrozza et al. [16] describe the SVEVIA framework for software quality assessment, decision support, and productivity management, which incorporates SRGM and was demonstrated on a large-scale project in collaboration with industry partners. While these tools have made software reliability research more accessible, each contains a relatively small number of models and are closed source. This latter limitation prevents other researchers, graduate students, and other interested parties from learning from implementations and contributing additional models to compare them objectively. Moreover, these tools frequently depend on a particular operating system and programming language or spreadsheet tool, which require payment of a fee prior to use.

To overcome the limitation of existing tools, a free and open source Software Failure and Reliability Assessment Tool (SFRAT) [17] was developed to foster a community of researchers and users. The open source nature of the tool enables users to incorporate the methods into their software testing work flows and researchers to contribute related statistical predictions.

This paper provides an overview of the SFRAT and automated script to accelerate the application of the tool. The script outputs a report in portable document format (PDF) and several other standard formats, eliminating the need to work with the graphical user interface to manually prepare reports. The script conserves time and promotes standardized application of software reliability models and reporting for ongoing projects. The script includes a verbose option to help new users interpret report results. The illustrations apply the tool to NASA project data.

The remainder of the paper is organized as follows: Section 2 provides an overview of the SFRAT, while Section 3 describes the automated script. Section 4 states the tool's impact. Section 5 concludes with directions for future research.

2. Software description

The Software Failure and Reliability Assessment Tool is a free and open source tool developed to promote regular quantitative assessment of software reliability, improve communication of such assessments, and foster dialog between the research and practitioner communities. The SFRAT implements early yet popular software reliability growth models, which employ techniques from statistics to identify a curve of best fit to failure data in order to predict future trends. Inferences enabled include the number of failures that would be detected with additional testing, how much the failure intensity will decrease as well as how much the mean time to failure and reliability will increase. Such increases are complemented by goodness of fit measures to objectively determine which of multiple models may predict best in order to guide release planning. For example, Major Defense Acquisition programs specify failure intensity as a Key System Attribute, which is a minimally acceptable value and therefore a threshold developers must achieve to ensure the system is operationally effective and suitable at the Full-Rate Production Decision Review. Passing this stage authorizes entry into the Full-Rate Production and is accompanied by the commitment of substantial taxpayer dollars.

The SFRAT is implemented in the R programming language as well as Python and can be used on computers running Windows, OSX, or Linux. The R source code is accessible on GitHub and runs in RStudio or similar R environment. The Python version runs with an interface such as Anaconda. The open source nature of the tool simplifies information assurance prior to use on sensitive failure data. Moreover, a web instance is available from the project website as well as example failure data sets. This eliminates the need to install software or format an organization's failure data in the required format before evaluation, enabling interested individuals to assess the tool's functionality in order to determine if it is suitable for their needs before investing additional time.

The initial SFRAT release supports three failure data formats, including interfailure time, failure time, and failure count as well as two trend tests for reliability growth to determine if application of models is appropriate. The SFRAT presently includes two hazard rate models, including the Jelinski–Moranda [18] and geometric [19], as well as four failure counting models, namely the Goel–Okumoto [20], delayed S-shaped [21], inflexion S-shaped [22], and Weibull [23]. Two measures of goodness of fit enable model selection. The application architecture has been designed to support incorporation of additional models and measures of goodness of fit into the tool. Okamura and Dohi [24] have contributed eleven models.

2.1. Select and analyze data

Fig. 1 shows the initial view of the SFRAT after loading the NASA1 data set [25], which consists of 60 unique failures observed over 168 units of test time. Input file formats include an Excel spreadsheet (.xlsx) or a CSV (comma separated value) (.csv). The Excel format permits one data set per worksheet, whereas CSV can only accommodate a single data set. The Select Sheet combo box allows the user to switch between data sets, which updates the graph. The Failure Data View Mode combo box enables visualization of the cumulative failures, time between failures, and failure intensity, the latter of which is defined as the reciprocal of the time between two successive failures.

Fig. 2 shows the Laplace Trend Test [26], which is a statistical test of reliability growth (increasing time between failures).

Data should exhibit reliability growth because software reliability growth models assume that the rate at which faults

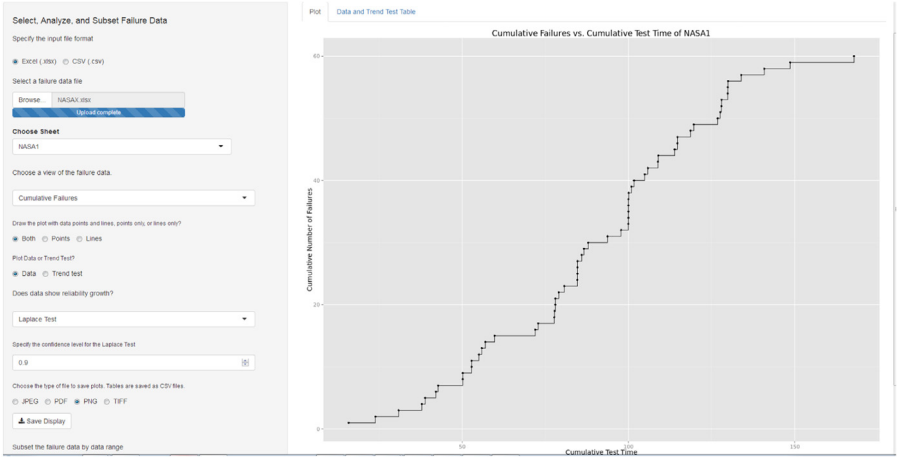

Fig. 1. Tab one view after upload.

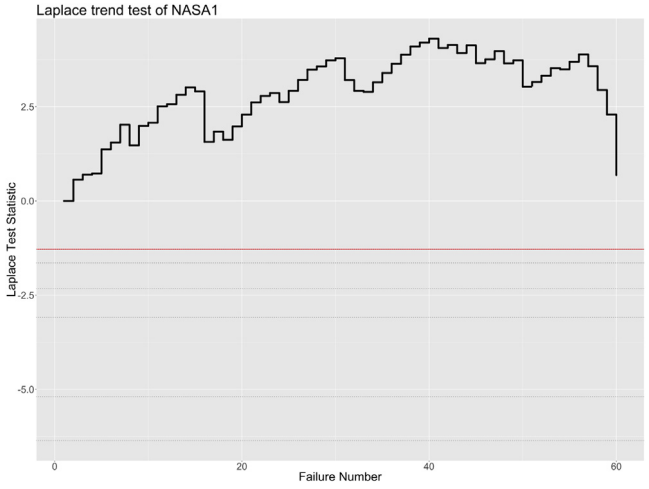

Fig. 2. Laplace trend test.

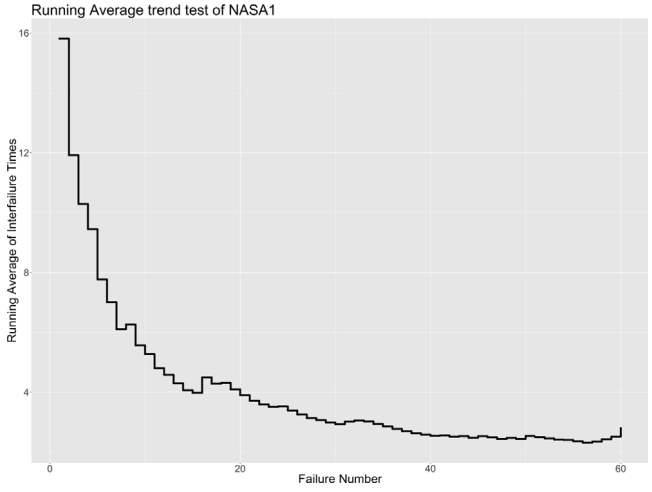

Fig. 3. Running arithmetic average.

are detected toward the end of testing decreases because fewer unique faults are discovered. If reliability growth is not present, applying models may be inappropriate because the predictions can be inaccurate. The value shown in Fig. 2 indicates the 90% significance (confidence) level in red, which has been specified by the user. Additional default levels in black include 90, 95, 99, and 99.9. 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. 3 shows the running arithmetic average of the NASA1 data set. If the time between failures increases, the running arithmetic average then increases, indicating an increase in system reliability. Similarly, a decreasing running arithmetic average indicates reliability deterioration. Both the Laplace trend test and running arithmetic average suggest that the example NASA1 data starts to trend in the desired direction toward the end of testing, which corresponds to the leveling off in the failure count to the right in Fig. 1.

Selecting the Table tab above any plot displays the raw numerical data used to produce the figure in a tabular format. The save icon below plots supports **.jpeg**, **.pdf**, **.png**, and **.tiff** formats for inclusion in reports, while numerical tables can be saved as a CSV or PDF file.

2.2. Set up and apply models

To apply models, the user selects the names of one or more model in the list on the left of Fig. 1 and then clicks compute. This executes the most complex algorithms contained in the software, namely numerical methods [27] to compute the maximum likelihood estimate or curve of best fit to the data.

Fig. 4 shows a plot of the inflexion S-shaped model superimposed on the data. The black vertical dashed line indicates the time at which the last failure was observed. Thus, points to the right of this line indicate prediction.

Figs. 5, 6, and 7 respectively show the times between failures, failure intensity, and reliability growth curve (probability of zero failures in a specified time interval) as well as with the corresponding model fits. The reliability growth curve does not include data because it is an average based on the model fitted to the data not a measure that can be plotted directly from the data.

2.3. Query model results

Models enable several practical inferences such as the time to achieve a specified reliability, expected number of failures in a specified time interval, and expected time until a specified number of future failures.

Fig. 8 shows the table of results for three models.

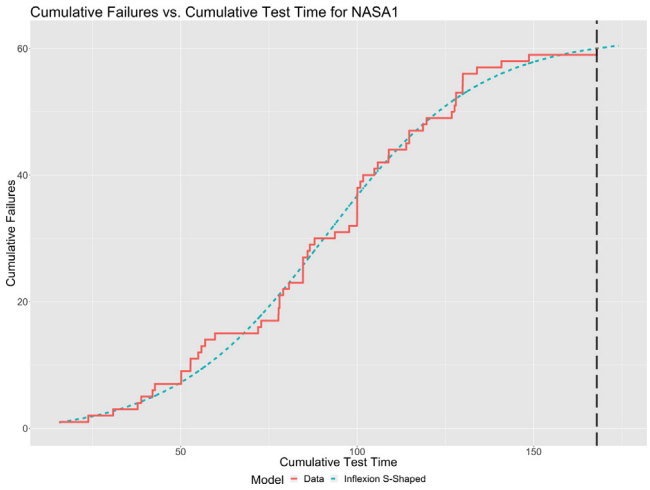

Fig. 4. NASA1 cumulative failures and model fits.

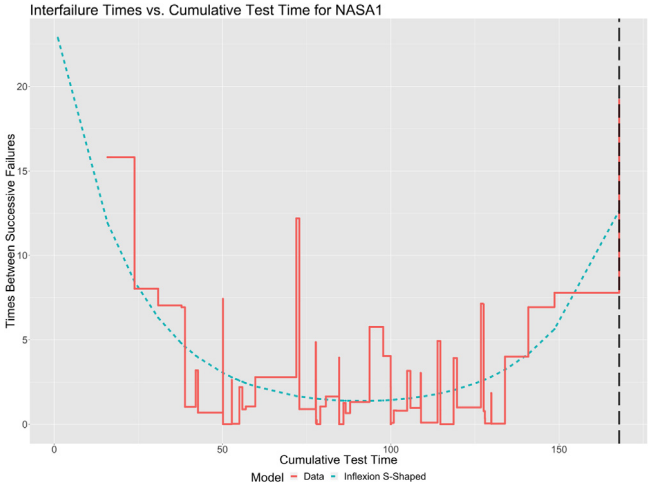

Fig. 5. NASA1 time between failures and model fit.

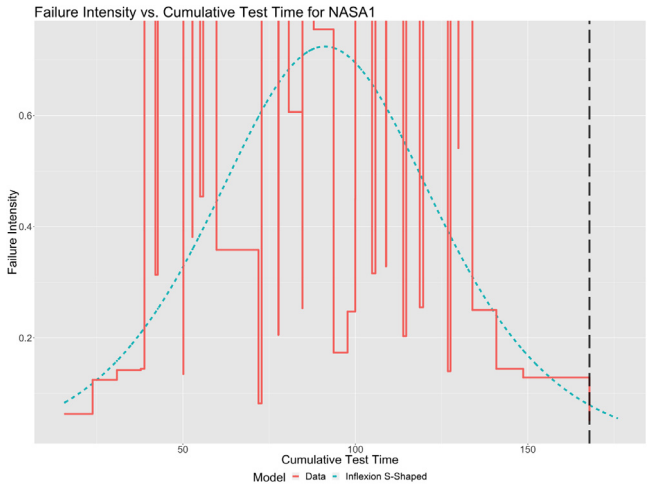

Fig. 6. NASA1 failure intensity and model fit.

2.4. Model evaluation

A natural question is which model to use for prediction. Therefore, the tool applies goodness of fit measures, including the Akaike information criterion (AIC) [28] and predictive sum of squares error (PSSE) [29], both of which prefer lower numerical values.

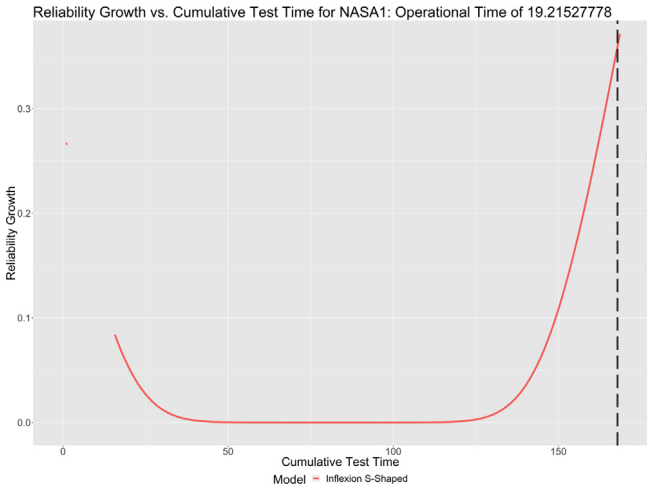

Fig. 7. NASA1 reliability growth curve.

	Model	Time to achieve R = 0.9 for mission of length 19.21527778	Expected # of failures for next 19.21527778 time units	Nth failure	Expected times to next 1 failures
	All	All	All	All	All
1	Delayed S-Shape	397.04584	5.423841	1	3.349118
2	Goel-Okumoto		6.868078	1	2.797767
3	Inflexion S-Shaped	49.98788	1.023417	1	18.529592

Fig. 8. Failure predictions.

Table 1
Goodness of fit measures models.

Model	AIC	PSSE
Inflexion S-shaped	223.00	41.85
Delayed S-Shaped	232.59	689.91
Goel-Okumoto	247.46	110.96

Table 1 shows goodness of fit measures for three models. The inflexion S-shaped model achieves the best scores with respect to AIC and PSSE suggesting that predictions based on this model may be more appropriate. It also indicates that the model performs well in a statistical sense not just the visual fit attained in Fig. 4.

3. Automated SFRAT report generation

While the Software Failure and Reliability Assessment Tool offers a user-friendly interface, each session requires time. In large software engineering projects, regular assessments will consume a fraction of an individuals time. To reduce time requirements, support standardization, and promote wider adoption, a script was developed in the R programming language to automatically generate reports in **pdf** and other common document formats. The user configures the script **report-specifications.R** by specifying values they would select via the user interface and the R Markdown file **SFRATReport.Rmd** is used in the graphical report generation process. This script can be incorporated into an organization's test and reporting process to produce consistent documentation in a standard format.

3.1. Script parameters

This section describes the parameters defined in the script, including variables that would otherwise be specified via the user interface.

The following is an example of the variables used to generate a report for the NASA1 data set, which is the first sheet of **NASAX.xlsx**.

```
verbose_report <- TRUE
sheetNumber <- 1
filePath <- '/SFRAT/model_testing/NASAX.xlsx'
colors <- c("navy","red","green","firebrick4","magenta")
confidence_lvl <- 0.9
num_failures_future_prediction <- 1
models_to_apply <- c('DSS', 'GM', 'Wei', 'GO', 'JM',
    'ISS')
mission_time <- 19.215
num_failures_to_predict <- 1
additional_time_software_will_run <- 19.215
desired_reliability <- 0.9
reliability_interval_length <- 19.215
percent_data_for_PSSE <- 0.9
```

Inline comments are supported, but have been removed here to conserve space.

The meaning of these parameters is as follows:

- **verbose_report** - Enabling this option produces verbose text output in the report, providing a brief description of each result in the report. This setting is appropriate for users unfamiliar with software reliability and the corresponding statistical inferences.
- **filepath** - The path to the location where the data file is located.
- **sheetNumber** - If the Excel file has more than one data set (one data set per sheet), this option allows the user to specify the data to be analyzed by indicating the sheet number.
- **colors** - The set of colors used to plot different models in the report.

3.2. Tab 1 - 'Select, Analyze, and Filter Data'

- **confidence_lvl** - A confidence level between 0 and 1 for the Laplace trend test (red line in Fig. 2) to quantify the desired level of significance for reliability growth.

3.3. Tab 2 - 'Set up and Apply Models'

- **num_failures_future_prediction** - Specify the number of failures to predict beyond the end of testing.
- **models_to_apply** - Specify which software reliability growth models to apply. The available models are the delayed s-shape (DSS), geometric (GM), Goel-Okumoto (GO), Jelinski-Moranda (JM), Weibull (Wei), inflexion S-shaped (ISS).
- **mission_time** - Extends the plot axis this amount of time beyond the last observed failure to display model trends into the future (for example, to the right of the vertical dotted line in Fig. 4).

3.4. 'Query Model Results'

- **num_failures_to_predict** - The number of failures to predict beyond the end of testing.
- **additional_time_software_will_run** - The additional time beyond the end of testing to predict the number of future failures.

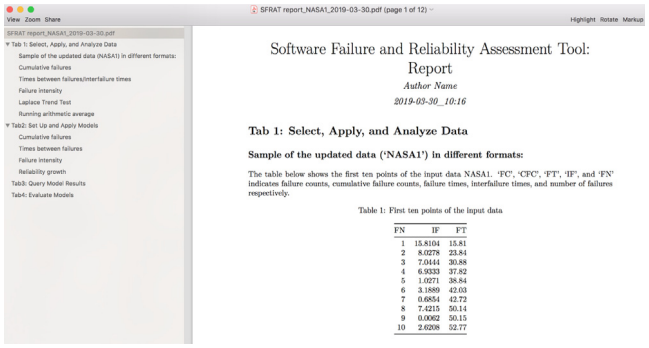

Fig. 9. Initial view of the report using NASA1 data.

- **desired_reliability** - The target reliability between 0 to 1 used to estimate the time required to achieve such reliability.
- **reliability_interval_length** - The duration the software should execute without failure to estimate reliability.

3.5. 'Evaluate Models'

- **percent_data_for_PSSE** - Percentage of data to be used for model fitting. The remaining data is used to assess model prediction using the predictive sum of squares error goodness-of-fit measure.

3.6. Example report

Fig. 9 shows the first page of the 12-page report generated.

The left side of the report provides an outline, which corresponds to the plots produced in a typical run of the workflow defined by the graphical user interface, including plots of the cumulative failures, time between failures, failure intensity, trend tests, and plots with the fitted models superimposed as well as the tables for model predictions and goodness of fit assessments. The default file name format for report is **SFRAT report_dataName_YYYY-MM-DD.pdf**.

4. Impact

The Software Failure and Reliability Assessment Tool offers to promote communication between software reliability researchers and practitioners. Specifically, the open source nature of the tool promotes the inclusion of additional models and measures of goodness of fit as well as the opportunity to extend the tool to additional stages of the software lifecycle. Thus, researchers contributing to the tool can rapidly transition their results to intended audiences, while practitioners can share data to communicate practical software engineering challenges in order to foster novel modeling research for their benefit.

Past research has overemphasized modeling and a limited number of optimization problems based on these models. Moreover, most new models proposed in research articles are compared to a handful of simpler models, which are relatively easy to outperform with statistical measures of goodness of fit. Similarly, optimization problems based on these models such as effort allocation [30] and optimal release [31] have rarely if ever demonstrated how such methods can be used in an ongoing manner throughout the testing process as additional failure data becomes available. The tool does not remove this barrier entirely, but aspires to foster meaningful collaboration between practitioners and researchers. The tool offers a framework for a researcher to contribute one or more models. Instructions are

available at https://sasdlc.org/lab/assets/projects/srt.html. These contributions will benefit users who can compare the performance of alternative models, while model contributors can gain credibility when their models are successfully applied to real projects.

The tool automates the complex mathematics required to identify curves of best fit to input data for each model with stable numerical methods, allowing users to focus on their day to day work. Thus, the software substantially lowers the barrier to applying software reliability engineering to quantitatively assess software and the automated script further reduces the time required to apply the methods, promoting standardization and regularity of reporting within and across projects. Toward this end, the source code can be modified and incorporated into internal processes of an organization.

The SFRAT is presently used by Federally Funded Research and Development Centers and Department of Defense University Affiliated Research Centers to assess the reliability of software intensive Major Defense Acquisition Programs to ensure that their government customers and the taxpayers receive quality software in support of national defense and security. The software is also used by major defense contractors, private industry, and the international research community (including a version with a Japanese language interface). However, we do not require individuals to register prior to download and therefore cannot accurately state the full extent of its use.

5. Conclusion and future work

This paper presented the Software Failure and Reliability Assessment Tool. The SFRAT supports analysis of failure data produced during testing, including trend tests for reliability growth, visualization of cumulative failures, time between failures, and failure intensity as well as reliability growth. To apply software reliability growth models to data, stable numerical methods have been implemented, enabling reliability predictions and goodness of fit assessment. The illustrations were provided in the context of a NASA data set and a script to further automate and standardize the software reliability assessment process was discussed. The tool is open source to promote adoption of the methods and support extensibility as well as further dialog between researchers and practitioners. It is presently used on large software applications acquired or developed by government and private organizations.

Future work will explore applications to software security defect (vulnerability) data discovered during penetration testing as well as integration with other software development and testing tools.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This material is based upon work supported by the National Aeronautics and Space Administration (NASA), USA Office of Safety and Mission Assurance (OSMA), USA Software Assurance Research Program (SARP), USA under Grant Number (#80NSSC18K0154) and the National Science Foundation, USA under Grant Number (#1749635). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NASA or the National Science Foundation.

The authors would also like to acknowledge the substantial contributions of Allen Nikora of the NASA Jet Propulsion Laboratory to the SFRAT.

References

[1] Jones C. A guide to selecting software measures and metrics. CRC Press; 2017.
[2] Lyu M, editor. Handbook of software reliability engineering. New York, NY: McGraw-Hill; 1996.
[3] Standard glossary of software engineering terminology (STD-729-1991). Tech. rep., ANSI/IEEE; 1991.
[4] Hudepohl J, Aud S, Khoshgoftaar T, Allen E, Mayrand J. Emerald: Software metrics and models on the desktop. IEEE Softw 1996;13(5):56–60.
[5] Software reliability modeling programs, Version 1.0. Tech. rep., Reliability and Statistical Consultants Ltd.; 1988.
[6] Draft software reliability engineering: Reliability estimation tools reference guide Version 3.7. Tech. rep., AT&T Bell Lsorelaboratories; 1990.
[7] Kanoun K, Kaaniche M, Laprie J, Metge S. SoRel: A tool for reliability growth analysis and prediction from statistical failure data. In: IEEE international symposium on fault-tolerant computing. 1993, p. 654–9.
[8] Farr W, Smith O. Statistical modeling and estimation of reliability functions for software (SMERFS) users guide. Tech. rep., Dahlgren, VA: Naval Surface Warfare Center; 1984, NAVSWC TR-84-373, Rev. 2.
[9] Lyu M, Nikora A. CASRE: A computer-aided software reliability estimation tool. In: IEEE international workshop on computer-aided software engineering. 1992, p. 264–75.
[10] Li N, Malaiya Y. ROBUST: A next generation software reliability engineering tool. In: IEEE international symposium on software reliability engineering. 1995, p. 375–80.
[11] Ramani S, Gokhale S, Trivedi K. SREPT: Software reliability estimation and prediction tool. Perform Eval 2000;39(1–4):37–60.
[12] Huang C-Y, Lyu M. Estimation and analysis of some generalized multiple change-point software reliability models. IEEE Trans Reliab 2011;60(2):498–514.
[13] Okamura H, Dohi T. SRATS: Software reliability assessment tool on spreadsheet (Experience report). In: International symposium on software reliability engineering. 2013, p. 100–7. http://dx.doi.org/10.1109/ISSRE.2013.6698909.
[14] Shibata K, Rinsaka K, Dohi T. M-SRAT: Metrics-based software reliability assessment tool. Int J Perform Eng 2015;11(4):369–79.
[15] Cinque M, Cotroneo D, Pecchia A, Pietrantuono R, Russo S. Debugging-workflow-aware software reliability growth analysis. Softw Test Verif Reliab 2017;27(7). e1638.
[16] Carrozza G, Pietrantuono R, Russo S. A software quality framework for large-scale mission-critical systems engineering. Inf Softw Technol 2018;102:100–16.
[17] Nagaraju V, Katipally K, Muri R, Wandji T, Fiondella L. An open source software reliability tool: A guide for users, in: International conference on reliability and quality in design, CA, 2016, p.132-7.
[18] Jelinski Z, Moranda P. Software reliability research. In: Statistical computer performance evaluation. Elsevier; 1972, p. 465–84.
[19] Moranda P. Prediction of software reliability during debugging. In: Annual reliability and maintainability symposium. 1975.
[20] Goel A. Software reliability models: Assumptions, limitations, and applicability. IEEE Trans Softw Eng 1985;(12):1411–23.
[21] Yamada S, Ohba M, Osaki S. S-shaped software reliability growth models and their applications. IEEE Trans Reliab 1984;33(4):289–92.
[22] Ohba M, Osaki S. Inflexion S-shaped software reliability growth models, stochastic models in reliability theory. Springer Berlin; 1984.
[23] Yamada S, Osaki S. Reliability growth models for hardware and software systems based on nonhomogeneous poisson process: A survey. Microelectron Reliab 1983;23(1):91–112.
[24] Okamura H, Dohi T. Srats: Software reliability assessment tool on spreadsheet (Experience report). In: IEEE international symposium on software reliability engineering (ISSRE). 2013, p. 100–7.
[25] Sukhwani H, Alonso J, Trivedi K, Mcginnis I. Software reliability analysis of nasa space flight software: A practical experience. In: IEEE International conference on software quality, reliability and security (QRS). 2016, p. 386–97.
[26] Gaudoin O. Optimal properties of the laplace trend test for soft-reliability models. IEEE Trans Reliab 1992;41(4):525–32.
[27] Burden R, Faires J. Numerical analysis. eighth ed.. Belmont, CA: Brooks/Cole; 2004.
[28] Akaike H. A new look at the statistical model identification. IEEE Trans Automat Control 1974;19(6):716–23.
[29] Fiondella L, Gokhale SS. Software reliability model with bathtub-shaped fault detection rate. In: Annual reliability and maintainability symposium. 2011, p. 1–6.
[30] Yamada S, Ohtera H, Narihisa H. Software reliability growth models with testing-effort. IEEE Trans Reliab 1986;R-35(1):19–23.
[31] Okumoto K, Goel A. Optimum release time for software systems based on reliability and cost criteria. J Syst Softw 1979;1:315–8.