A New Statistical Software Reliability Tool

M.A.A. Boon[1], E. Brandt[2], I. Corro Ramos[1], A. Di Bucchianico[1] and R. Henzen[2]
[1] *Department of Mathematics, Eindhoven University of Technology, Eindhoven, The Netherlands*
[2] *Refis System Reliability Engineering, Bilthoven, The Netherlands*

Abstract

We describe a new statistical tool for software reliability analyses that we are developing. Existing packages for statistical analysis of software reliability data do not make full use of state-of-the-art statistical methodology or do not conform to best practices in statistics. Our tool has a Java based interface and uses the statistical programming language `R` (see `www.r-project.org`) for the statistical computations. `R` is open-source free software maintained by a group of top-level statisticians and is rapidly becoming the standard programming language within the statistical community. The tool has a user-friendly interface which includes features like auto detection of data type and a model selection wizard.

Keywords: software reliability, software testing, statistical models, `R`.

1 Introduction

Successful testing processes require excellence in both software testing and management. In order to support well-founded decisions on issues like resource allocation and software release moments, quantitative procedures are indispensable. Since few testing processes have a deterministic course, statistics is very often an appropriate part of such quantitative procedures. Existing tools for software reliability analysis like Casre and Smerfs[3] do not make full use of state-of-the-art statistical methodology or do not conform to best practices in statistics. Thus, these tools cannot fully support sound software reliability analyses. We decided to build a new tool that

- uses well-documented state-of-the-art algorithms
- is platform independent
- encourages to apply best practices from statistics
- can easily be extended to incorporate new models.

In order to meet these requirements we decided to use Java for the interface and the statistical programming language `R` (see `www.r-project.org`) for the statistical computations. `R` is open-source free software maintained by a group of top-level statisticians and is rapidly becoming the standard programming language within the statistical community. In this paper we report on the status of our tool. Our tool is a joint project of the Laboratory for Quality Software (LaQuSo) of the Eindhoven University of Technology (`www.laquso.com`) and Refis (`www.refis.nl`). The tool development is financially supported by a grant of the Dutch Innovation Platform.

The rest of the paper is organized as follows. In Section 2 general guidelines of software reliability analysis are given. In Section 3 we present general and statistical features of the tool. We focus on describing the tool's GUI thoroughly. A study to show how the tool works in practice is presented in Section 4. Finally, in Section 5 we summarize the work carried out and we point out to several important short-term objectives.

2 A note on reliability analysis

For general information on software reliability analysis we refer to Lyu (1996), Musa (2006) and Pham (2006). Like there exist coding standards for writing software, there also exist standards for performing statistical analyses. Basic steps in a statistical analysis of software reliability data should include (cf. Goel (1985))

1. data collection (which data is relevant for the analysis)
2. trend tests (does the data indicate growth, otherwise analysis is useless)
3. model selection (pre-selection of models)
4. model estimation (calculate optimal parameters from data)
5. model validation (do models fit to data)
6. model interpretation (calculate quantities of interest from model parameters).

We talk of ungrouped or exact data when the failures are reported individually and the data represents time between failures. However, it is possible to report faults in periods of time, in which case the data consists of the time intervals where the failures are reported and the number or faults found in each interval. In this case we talk of grouped or interval data. Unfortunately, ungrouped data has received much more attention in the literature.

Before trying to fit any reliability growth model we should verify whether the data indicates reliability growth. This can be done using adequate plots or more formally with trend tests. Figure 1 clearly depicts the idea that software becomes more reliable as long as it is tested and errors have been repaired, so that more effort is required to find future errors.

There are over 200 software reliability models based on different assumptions, assumptions which are often unclear or too unrealistic. Systematic approaches to use model assumptions and data requirements for initial model selection have not received much attention in the literature, Kharchenko et al. (2002) being an exception. Therefore, we have been developing a matrix-based procedure to support the choice of the models to work with. A simple version of this matrix can be found in Figure 2. Since we wish to select rather than rule out applicable models, we state all assumptions as negations of restrictions. To select models, one first has to select relevant assumptions and weights to incorporate the available information on the testing project at hand. If all requirements and selected assumptions of a model are satisfied, then the score for this model is 100%. In all other cases the score of the model is defined using the relative importance (weight) of the applicable characteristics of the model.

Estimation of model parameters requires optimization. Since the parameters typically are of different order of magnitude, numerical problems like non-convergence or large flat areas

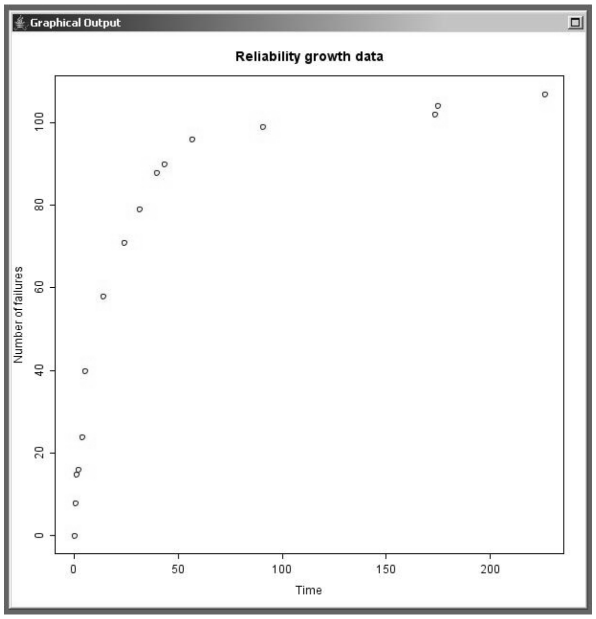


Figure 1: Reliability growth data.

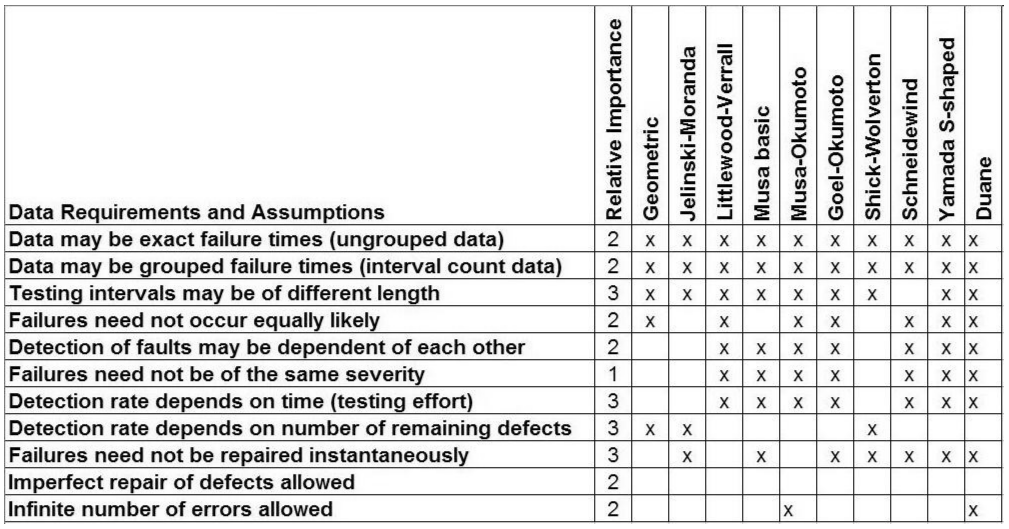
around the maximum cause practical problems (see Yin and Trivedi (1999)). In our tool we pay attention to convergence issues and apply algorithms that avoid the standard numerical problems.

After parameter estimation has been completed, model verification must be determined. Graphical methods like the u-plot or TTT (Total Time on Test) plots (see Rigdon and Basu (2000) for details) or goodness-of-fit tests can be applied. Unluckily we find also problems here (derived to the fact that the assumptions of independent and identically distributed observations are normally broken by software reliability models). Therefore, standard goodness-of-fit tests, like the Kolmogorov test, cannot be used although this is often done. For diverse subclasses of models new goodness-of-fit tests are becoming available (see Bhattacharjee et al. (2004), Zhao and Wang (2005)).

Finally, model interpretation makes possible to determine quantities like number of remaining errors and reliability of the system.

3 Software reliability tool

In this section we give a general overview of the GUI of our tool. Our tool is written in Java and we have made use of readily available components from Java Resource Bundles. The statistical computations are performed by calling R (high-quality free open-source statistical software). The communication of Java with R uses JRI and JavaGD libraries developed by RoSuDa, the Computational Statistics group of the University of Augsburg. Initially, the GUI of our tool consists of four menu items as we can see in Figure 3. The multiple options



Data Requirements and Assumptions	Relative Importance	Geometric	Jelinski-Moranda	Littlewood-Verrall	Musa basic	Musa-Okumoto	Goel-Okumoto	Shick-Wolverton	Schneidewind	Yamada S-shaped	Duane
Data may be exact failure times (ungrouped data)	2	x	x	x	x	x	x	x	x	x	x
Data may be grouped failure times (interval count data)	2	x	x	x	x	x	x	x	x	x	x
Testing intervals may be of different length	3	x	x	x	x	x	x	x		x	x
Failures need not occur equally likely	2	x		x		x	x		x	x	x
Detection of faults may be dependent of each other	2			x	x	x	x		x	x	x
Failures need not be of the same severity	1			x	x	x	x		x	x	x
Detection rate depends on time (testing effort)	3			x	x	x	x		x	x	x
Detection rate depends on number of remaining defects	3	x	x					x			
Failures need not be repaired instantaneously	3		x		x		x	x	x	x	x
Imperfect repair of defects allowed	2										
Infinite number of errors allowed	2					x					x

Figure 2: Assumption matrix.

of all the menus are explained during this section. In the beginning only the *Data* and *Help* menus are enabled. To have access to the working environment we have to select the option *Import* from the *Data* menu. We can distinguish two new windows that will remain visible all the time, one devoted to data and the other one to graphics. Figure 4 shows the data window. Three different tabs can be identified. The first one contains the imported data, the second one is allocated to the filtered data (in case we use this option) and the third one will show the data from the model analysis. Figure 1 shows the *Graphical Output* window where all the plots produced will be displayed. In the rest of this section we explain the tool menu options in detail, paying special attention to the *Models* and *Analysis* menus.

3.1 Data menu

1. **Import**

   Imports data files in .txt and .xls formats. After loading the data set the working environment is displayed.

2. **Export**

   Exports data files in .txt and .xls formats.

3. **Filter Data**

   This dialog (see Figure 5) allows to refine the data set and keep a subset of observations. The filtered data set can also be exported and imported.

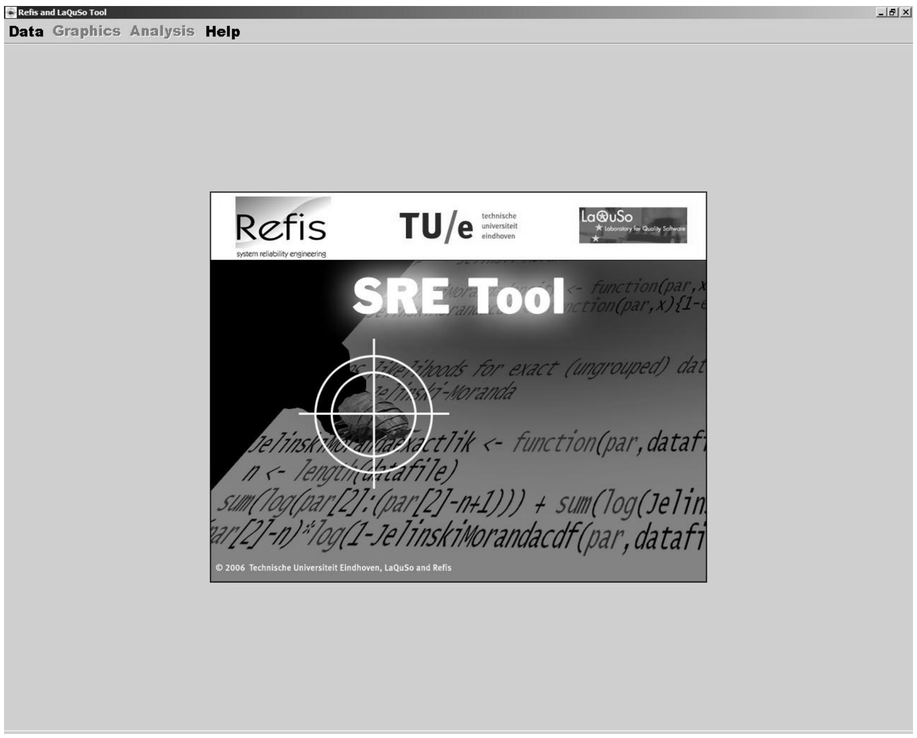
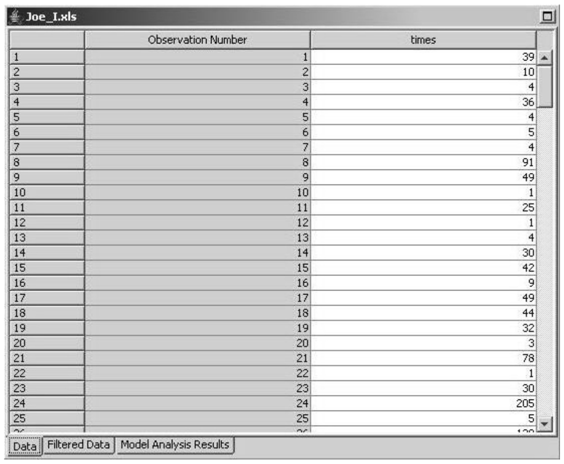

Refis and LaQuSo Tool
Data
Graphics
Analysis
Help
Refis
system reliability engineering
TU/e technische universiteit eindhoven
LaQuSo
SRE Tool
© 2006 Technische Universiteit Eindhoven, LaQuSo and Refis

Figure 3: Software reliability tool GUI.



	Observation Number	times
1	1	39
2	2	10
3	3	4
4	4	36
5	5	4
6	6	5
7	7	4
8	8	91
9	9	49
10	10	1
11	11	25
12	12	1
13	13	4
14	14	30
15	15	42
16	16	9
17	17	49
18	18	44
19	19	32
20	20	3
21	21	78
22	22	1
23	23	30
24	24	205
25	25	5

Figure 4: Data window.

4. **Transform Variable**

   With this option we can create new variables applying basic operations to the variables of the data set. The available operations can be observed in Figure 6.

5. **Preferences**

   User preferences can be set using this option. Font type and size can be chosen for both program GUI and output (see Figure 7). The number of significant digits for the calculations can be selected from one up to eight (by default set to three). We can also

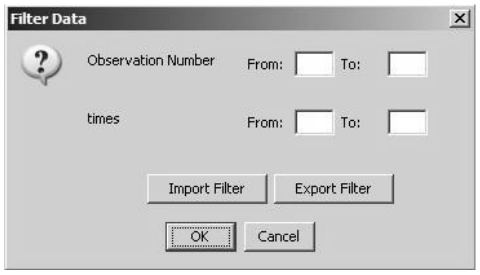


Figure 5: Filter Data dialog.

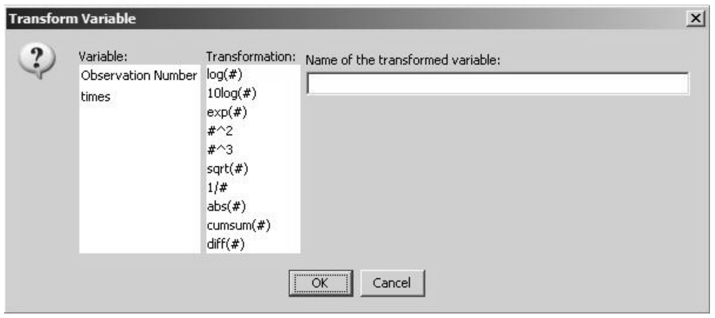


Figure 6: Transform Variable dialog.

decide the colours to be used in the plots. These options can be saved (or load) to (from) a .txt file.

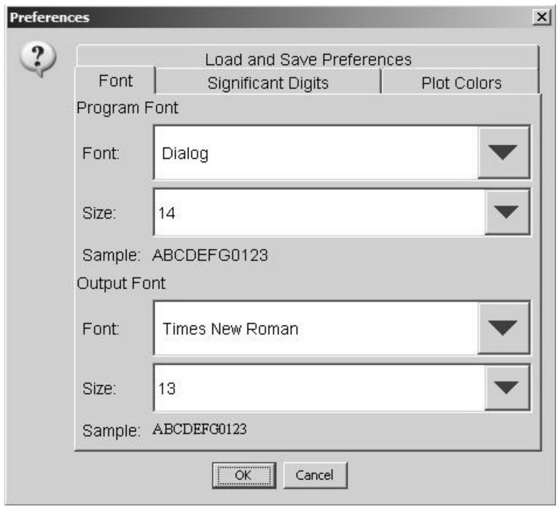
6. **Exit**

   Closes the GUI and terminates the program execution.

3.2 Graphics menu

1. **Create Scatter Plot**

   With this option we can generate plots that will be loaded in the *Graphical Output* window. After clicking on this item, a window named *Select Plot Variables* pops-up (see Figure 8). We can select the variables we want to plot and assign them to the X or Y axes. Moreover, we can choose the type of plot we wish to generate. The existing options are scatter plot, normal probability plot and TTT plot. Finally, the *Draw Graph* button will create the graph. On the lower part of the window we can set



Program Font

Font: Dialog

Size: 14

Sample: ABCDEFG0123

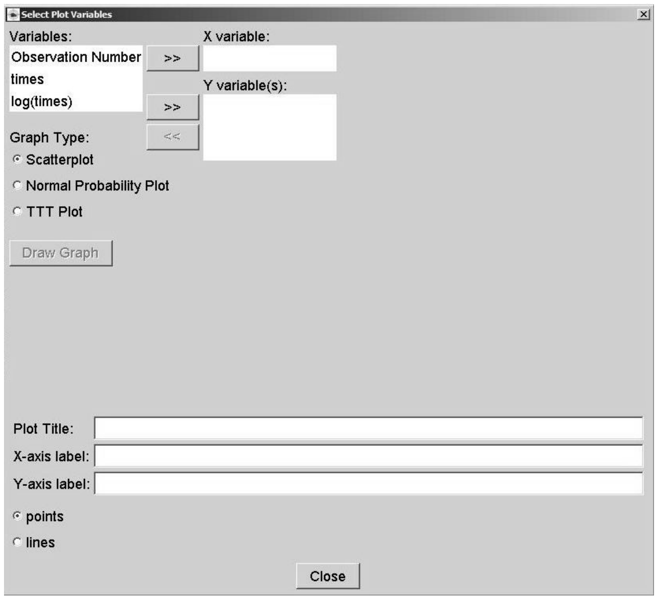
Output Font

Font: Times New Roman

Size: 13

Sample: ABCDEFG0123

Figure 7: Preferences window.



Variables: Observation Number, times, log(times)

X variable:

Y variable(s):

Graph Type: [✓ Scatterplot] Normal Probability Plot TTT Plot

Plot Title:

X-axis label:

Y-axis label:

[✓ points] lines

Figure 8: Select Plot Variables window.

display options of the plot like the main title, the labels of the axes or the format of the curves.

2. **Copy Current Plot to Clipboard**

   Copy the plot that is displayed on the *Graphical Output* window to the clipboard. Thus, the plot is cached and can be transferred between documents or applications, via copy and paste operations.

3. **Export Current Plot**

   Selecting this option a save dialog appears to export the current plot to a graphic file. The formats supported are .eps, .jpg and .png.

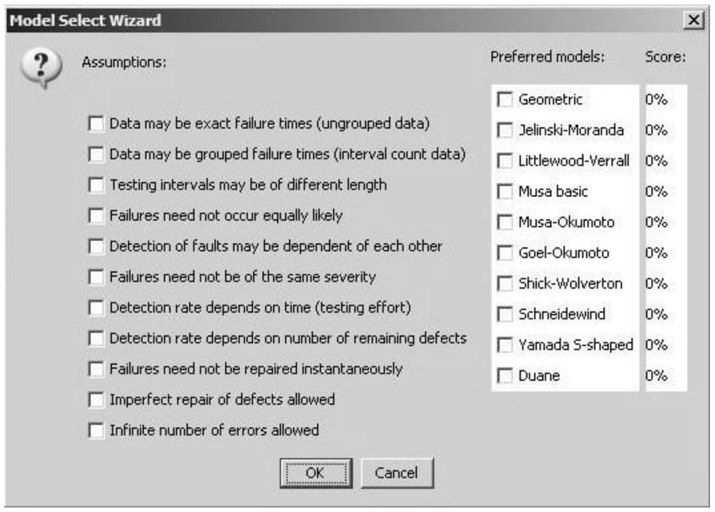
4. **Print Current Plot**

   With this option a print dialog is displayed to print the current plot.

3.3 Analysis menu

1. **Model Select Wizard**

   Figure 9 shows the *Model Select Wizard*. In this dialog a list of common software reliability assumptions are enumerated. These assumptions are known from the literature and they are used in different software reliability models. The most relevant models are listed on the right-hand side of the dialog follow by a column called *Score*. Every assumption has a score associated to every model. After selecting the assumptions, the models receive points and they are sorted by relevance. The higher score a model has, the better the model will fit the assumptions.



- [ ] Data may be exact failure times (ungrouped data)
- [ ] Data may be grouped failure times (interval count data)
- [ ] Testing intervals may be of different length
- [ ] Failures need not occur equally likely
- [ ] Detection of faults may be dependent of each other
- [ ] Failures need not be of the same severity
- [ ] Detection rate depends on time (testing effort)
- [ ] Detection rate depends on number of remaining defects
- [ ] Failures need not be repaired instantaneously
- [ ] Imperfect repair of defects allowed
- [ ] Infinite number of errors allowed

Preferred models:	Score:
☐ Geometric	0%
☐ Jelinski-Moranda	0%
☐ Littlewood-Verrall	0%
☐ Musa basic	0%
☐ Musa-Okumoto	0%
☐ Goel-Okumoto	0%
☐ Shick-Wolverton	0%
☐ Schneidewind	0%
☐ Yamada S-shaped	0%
☐ Duane	0%

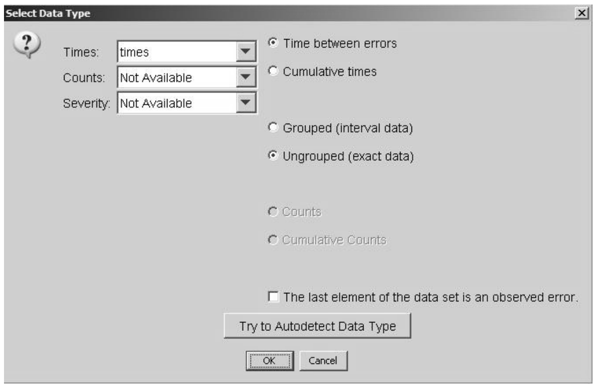
Figure 9: Model Select Wizard dialog.

2. **Data Type**

   With the *Select Data Type* window (see Figure 10) we have the option to set the type of data of our data set. Time between failures or cumulative times, exact or interval data are the available options. In case of interval data we have to distinguish between counts per interval or cumulative counts. Options like different kind of errors (severity) or whether the last element of the data set is an observer error are also offered. In case we do not know which kind of data we have, the option *Try to Autodetect Data Type* results of special interest. With this button the program estimates the type of data of our data file. Checking the number of columns (grouped or ungrouped) or the growth of the data (time between failures or cumulative times) we may find out what kind of data we are using.

3. **Trend Tests**

   The tool has the Laplace and the MIL-HDBK 189 trend tests already implemented and new tests will be added soon. The *Trend Tests* window is shown in Figure 11. Select



Times: times
Counts: Not Available
Severity: Not Available

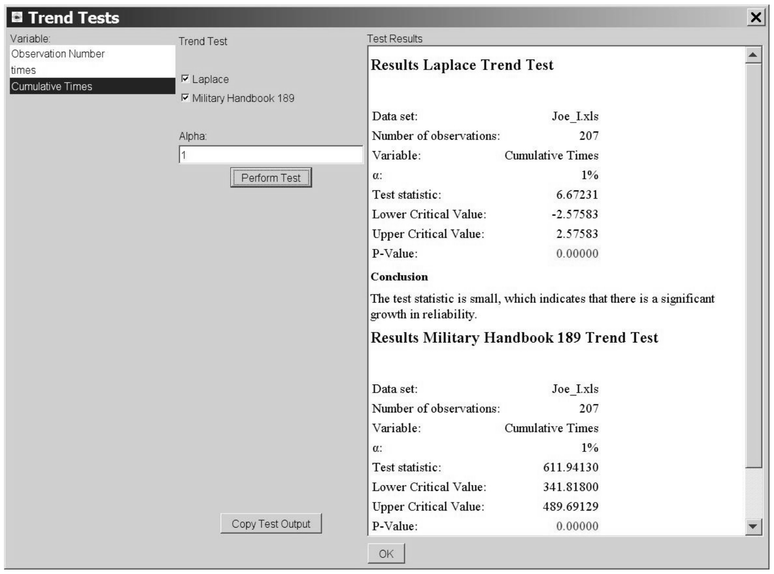
[✓ Time between errors] Cumulative times

Grouped (interval data) [✓ Ungrouped (exact data)]

Counts Cumulative Counts

☐ The last element of the data set is an observed error.

Figure 10: Select Data Type dialog.



Variable: Observation Number, times, [✓ Cumulative Times]

Trend Test: ☑ Laplace ☑ Military Handbook 189

Alpha: 1

Test Results

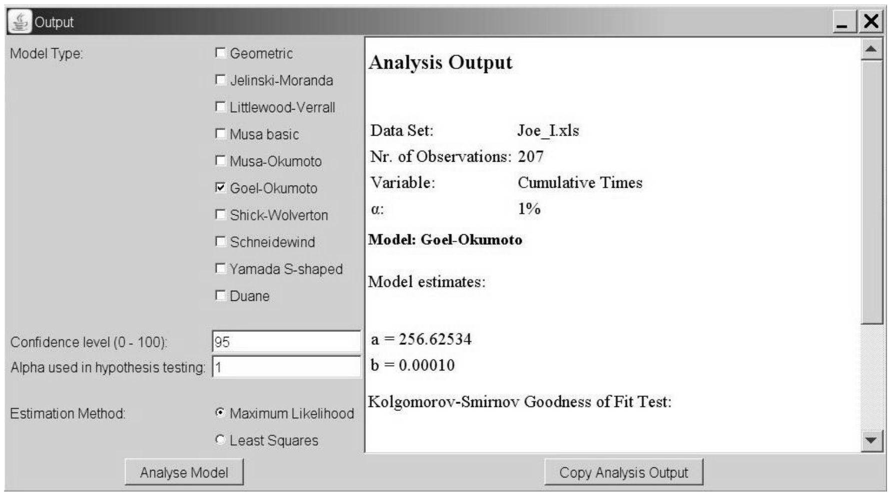
```
Results Laplace Trend Test

Data set:                    Joe_I.xls
Number of observations:            207
Variable:              Cumulative Times
α:                                  1%
Test statistic:                6.67231
Lower Critical Value:         -2.57583
Upper Critical Value:          2.57583
P-Value:                       0.00000
Conclusion
The test statistic is small, which indicates that there is a significant
growth in reliability.
Results Military Handbook 189 Trend Test

Data set:                    Joe_I.xls
Number of observations:            207
Variable:              Cumulative Times
α:                                  1%
Test statistic:              611.94130
Lower Critical Value:        341.81800
Upper Critical Value:        489.69129
P-Value:                       0.00000
```

Figure 11: Trend Tests window.

the variable whose growth we want to investigate as well as the tests we would like to use. The significance at which the test will be performed can also be selected here. On the right-hand side of the window we can observe the result of the tests we decided to perform. Graphical interpretation of the test is also displayed on the *Graphical Output* window.

4. **Analyse Model**

   The analysis window is shown in Figure 12. Before performing the analysis four steps must be follow. First, select the model(s) to fit the data. Then set the confidence level used to calculate confidence intervals (by default set to 95%) and the significance level to perform hypothesis testing (by default set to 1%). Last, choose between the maximum-likelihood or least squares methods for parameter estimation. To perform



Model Type: [ ] Geometric [ ] Jelinski-Moranda [ ] Littlewood-Verrall [ ] Musa basic [ ] Musa-Okumoto [✓ Goel-Okumoto] [ ] Shick-Wolverton [ ] Schneidewind [ ] Yamada S-shaped [ ] Duane

Confidence level (0 - 100): 95

Alpha used in hypothesis testing: 1

Estimation Method: [✓ Maximum Likelihood] Least Squares

Analysis Output

Data Set:	Joe_I.xls
Nr. of Observations:	207
Variable:	Cumulative Times
α:	1%

Model: Goel-Okumoto

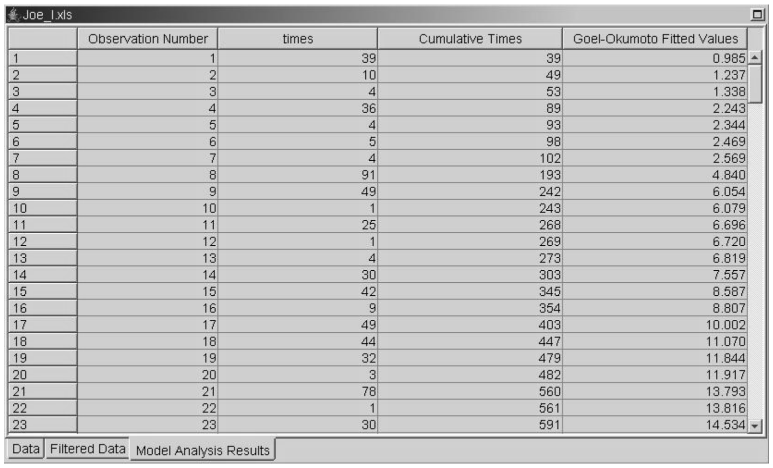
Model estimates:

a = 256.62534
b = 0.00010

Kolgomorov-Smirnov Goodness of Fit Test:

Figure 12: Analysis output.

the analysis click the *Analyse Model* button. Subsequently, the analysis output shows the type of data, the model choices, the parameter estimates and the result of the Kolmogorov goodness-of-fit test. The data window (see Figure 13) presents the fitted values for the selected models. Finally, the graphical output (see Figure 14) shows the



	Observation Number	times	Cumulative Times	Goel-Okumoto Fitted Values
1	1	39	39	0.985
2	2	10	49	1.237
3	3	4	53	1.338
4	4	36	89	2.243
5	5	4	93	2.344
6	6	5	98	2.469
7	7	4	102	2.569
8	8	91	193	4.840
9	9	49	242	6.054
10	10	1	243	6.079
11	11	25	268	6.696
12	12	1	269	6.720
13	13	4	273	6.819
14	14	30	303	7.557
15	15	42	345	8.587
16	16	9	354	8.807
17	17	49	403	10.002
18	18	44	447	11.070
19	19	32	479	11.844
20	20	3	482	11.917
21	21	78	560	13.793
22	22	1	561	13.816
23	23	30	591	14.534

Figure 13: Fitted values data.

observed values and the estimated models (by default represented by circles and a solid curve, respectively).

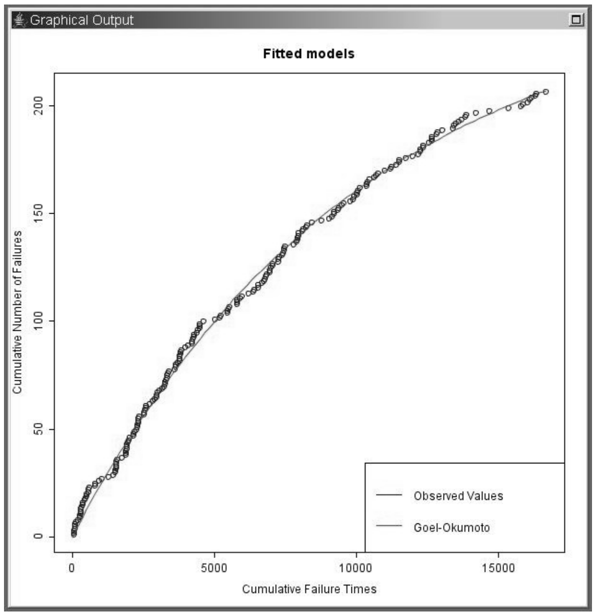


Figure 14: Fitted model plot.

5. **Plot Fitted Models**

   This option is enabled only after the analysis has been done and allows us to plot the observed values and the estimated models that we selected for the analysis.

6. **Export Output to Excel**

   This option is also enabled only after the analysis has been performed and give us the possibility to export the output of the analysis to .xls format.

3.4 Help menu

1. **Tool Help**

   Help files of the tool.

2. **SRE Help**

   Glossary of terms and background on chosen algorithms.

3. **About**...

   Information about the developers, version of the tool, etc...

4 Case study

In this section we show a demonstration of the tool using a data set by Joe (1989). The data set, reproduced in Table 1, consists of 207 observations corresponding to times between failures (ungrouped data). To load the data set in our tool click the *Data* menu and then on

Joe_ I data set													
39	10	4	36	4	5	4	91	49	1	25	1	4	30
42	9	49	44	32	3	78	1	30	205	5	129	103	224
186	53	14	9	2	10	1	34	170	129	4	4	35	5
5	22	36	35	121	23	33	48	32	21	4	23	9	13
165	14	22	41	12	138	95	49	62	2	35	89	99	69
22	15	19	42	14	11	41	210	16	30	37	66	9	16
14	24	12	159	89	118	29	21	18	2	114	37	46	17
1	150	382	160	66	206	9	26	62	239	13	4	85	85
240	178	34	102	9	146	59	48	25	25	111	5	31	51
6	193	27	25	96	26	30	30	17	320	78	39	13	13
19	128	34	84	40	177	349	274	82	58	31	114	39	88
84	232	108	38	86	7	22	80	239	3	39	63	152	63
80	245	196	46	152	102	9	228	220	208	78	3	83	6
212	91	3	10	172	21	173	371	40	48	126	90	149	30
317	500	673	432	66	168	66	66	128	49	332			

Table 1: Time between failures.

Import. An open dialog pops up (see Figure 15). We look for the data file that contains our data set (Joe_ I.xls) and then we click the *Open* button. We can distinguish three different windows in the GUI as we mentioned in the beginning of Section 3. On the left-hand side we can observe the data file while the graphical output window appears on the right-hand side. The middle of the screen is occupied by the *Select Data Type* dialog, already presented in Section 3.3 as part of the *Data Type* menu (see Figure 10). With this dialog we decide the kind of data we have. Therefore, select *Time between errors* and *Ungrouped* options. Note that in case of doubt we can try the option *Try to Autodetect Data Type.* The next step is to check whether the data presents any trend. Select the option *Trend Tests* from the *Analysis* menu and the *Trend Tests* window will be displayed (see Section 3.3, Trend Tests). Select the variable *Cumulative Times* follow by the trend tests we want to perform, in this case both Laplace and MIL-HDBK 189. Set the statistical significance level (*Alpha*) for the analysis and then click on *Perform Test.* The result of the tests is shown in Figure 11. On the *Test Results* area we can observe that both tests support the fact that there exists significant growth in reliability. In addition, the graphical output displays a graphical interpretation of the (statistical) test (in this case MIL-HDBK 189 test) as we can appreciate in Figure 16. Since the result of the trend test is positive, *i.e.*, the data shows reliability growth, the next step is to perform analysis of the data applying software reliability models. An initial model inspection can be done with the option *Model Select Wizard* from the *Analysis* menu, as we described in Section 3.3 (see Figure 9). Choose the most appropriate assumptions to our case and select the model (or models) with the highest score. In this case, Goel-Okumoto seems to be a suitable model for our assumptions, as we can appreciate in Figure 17. Select *Analyse Model*

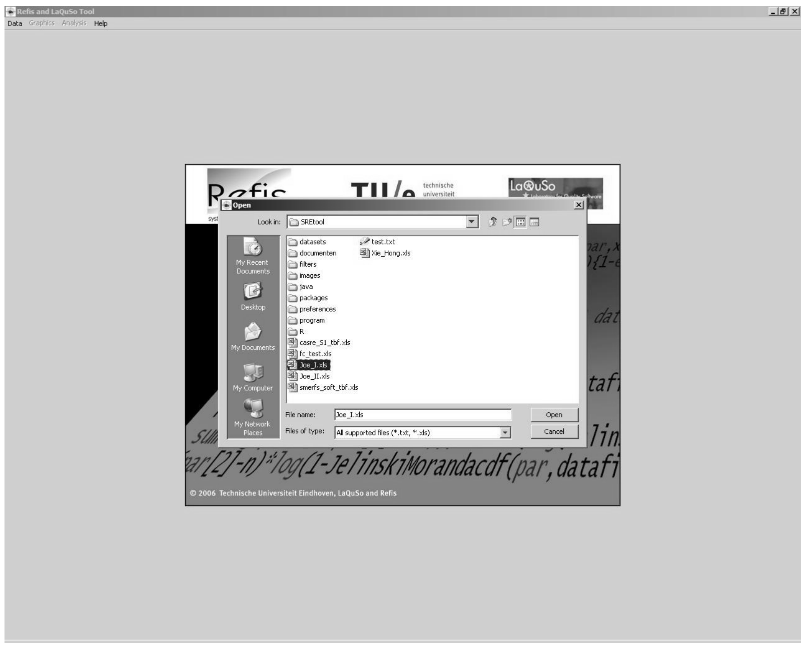


Figure 15: Browsing the data file.

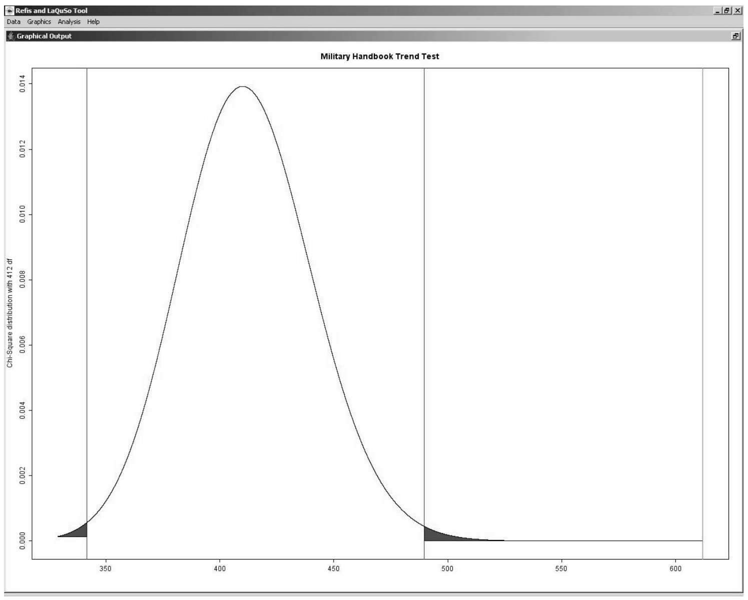
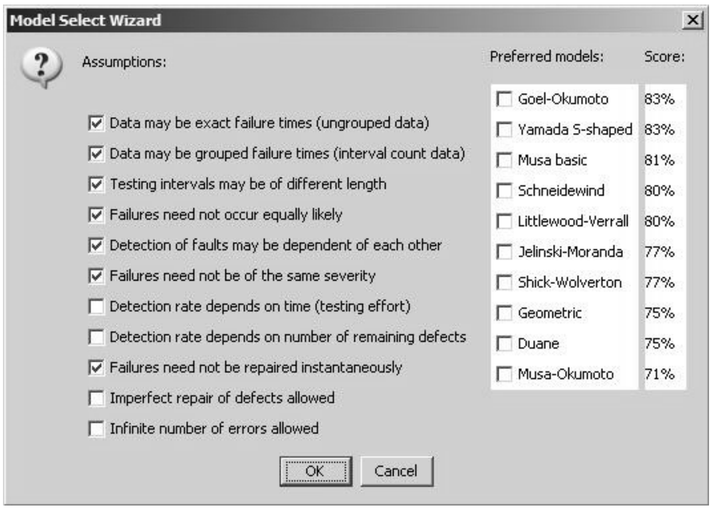


Figure 16: MIL-HDBK 189 trend test graphical interpretation.

from the *Analysis* menu and the analysis output window (see Figure 12) will be displayed. Select the Goel-Okumoto model and set the confidence level that will be used to calculate confidence intervals and the significance level to perform hypothesis testing. Choose between maximum likelihood or least squares parameter estimation methods and click on *Analyse Model*. The result of the analysis is shown in Figure 18. The data window presents the fitted values for the Goel-Okumoto model. The graphical output shows the observed values (circles)

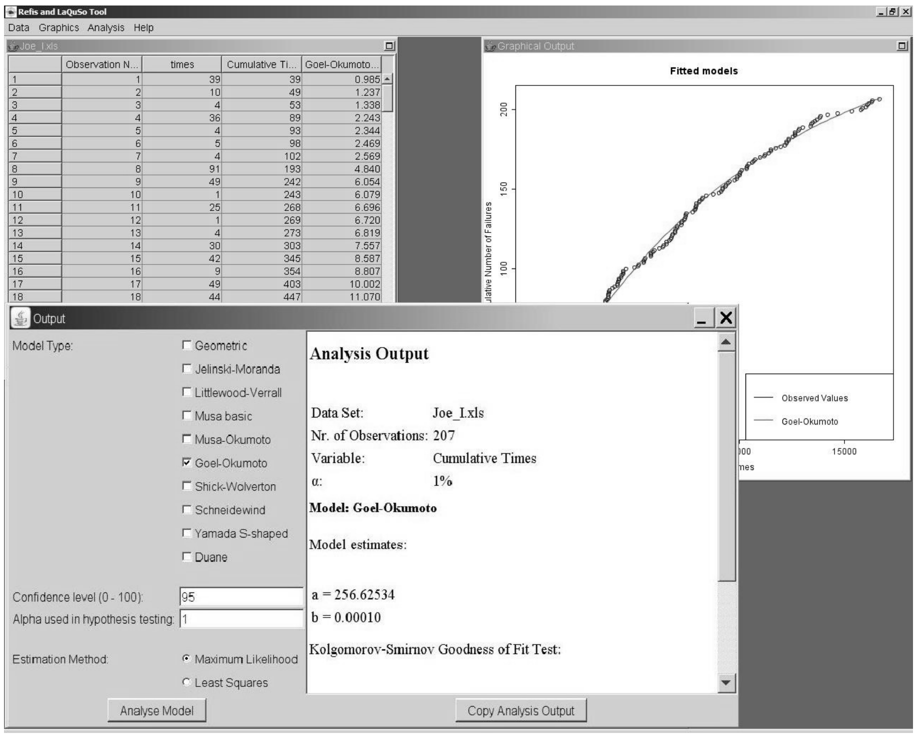


Assumptions:

- [✓] Data may be exact failure times (ungrouped data)
- [✓] Data may be grouped failure times (interval count data)
- [✓] Testing intervals may be of different length
- [✓] Failures need not occur equally likely
- [✓] Detection of faults may be dependent of each other
- [✓] Failures need not be of the same severity
- [ ] Detection rate depends on time (testing effort)
- [ ] Detection rate depends on number of remaining defects
- [✓] Failures need not be repaired instantaneously
- [ ] Imperfect repair of defects allowed
- [ ] Infinite number of errors allowed

Preferred models:	Score:
Goel-Okumoto	83%
Yamada S-shaped	83%
Musa basic	81%
Schneidewind	80%
Littlewood-Verrall	80%
Jelinski-Moranda	77%
Shick-Wolverton	77%
Geometric	75%
Duane	75%
Musa-Okumoto	71%

Figure 17: After selecting our assumptions, the Goel-Okumoto model gets the highest score.



	Observation N...	times	Cumulative Ti...	Goel-Okumoto...
1	1	39	39	0.985
2	2	10	49	1.237
3	3	4	53	1.338
4	4	36	89	2.243
5	5	4	93	2.344
6	6	5	98	2.469
7	7	4	102	2.569
8	8	91	193	4.840
9	9	49	242	6.054
10	10	1	243	6.079
11	11	25	268	6.696
12	12	1	269	6.720
13	13	4	273	6.819
14	14	30	303	7.557
15	15	42	345	8.587
16	16	9	354	8.807
17	17	49	403	10.002
18	18	44	447	11.070

Model Type:
- [ ] Geometric
- [ ] Jelinski-Moranda
- [ ] Littlewood-Verrall
- [ ] Musa basic
- [ ] Musa-Okumoto
- [✓] Goel-Okumoto
- [ ] Shick-Wolverton
- [ ] Schneidewind
- [ ] Yamada S-shaped
- [ ] Duane

Confidence level (0 - 100): 95
Alpha used in hypothesis testing: 1

Estimation Method: [✓ Maximum Likelihood] Least Squares

Analysis Output

```
Data Set:              Joe_I.xls
Nr. of Observations:   207
Variable:              Cumulative Times
α:                     1%
```

Model: Goel-Okumoto

Model estimates:

a = 256.62534
b = 0.00010

Kolgomorov-Smirnov Goodness of Fit Test:

Figure 18: Model analysis.

and the estimated model (curve). A first inspection of this plot suggests that Goel-Okumoto is a reasonable model for our data. Formal result of the analysis (parameter estimates and goodness-of-fit test) is displayed on the analysis output window. As a consequence of our analysis we can conclude that there is no evidence to reject the fact that our data can be described using the Goel-Okumoto model.

5 Conclusions and future work

We have developed a new tool for statistical software reliability analyses. The interface is programmed in Java and therefore, it is platform independent. Our tool uses well-documented state-of-the-art statistical algorithms programmed in R and encourages to apply best practices from statistics. Furthermore, it has a user-friendly interface which includes features like model selection wizard and auto detection of data type. The extension of the tool incorporating new models and features is an ongoing challenge that stimulates us to continue working on this direction.

References

M. Bhattacharjee, J. V. Deshpande, and U. V. Naik-Nimbalkar. Unconditional tests of goodness of fit for the intensity of time-truncated nonhomogeneous Poisson processes. *Technometrics*, 46(3):330–338, 2004.

A.L. Goel. Software reliability models: Assumptions, limitations, and applicability. *IEEE Trans. Soft. Eng.*, 11(12):1411–1423, 1985.

H. Joe. Statistical inference for General-Order-Statistics and Nonhomogeneous-Poisson-Process software reliability models. *IEEE Trans. Software Eng.*, 15(11):1485–1490, 1989.

V.S. Kharchenko, O.M. Tarasyuk, V.V. Sklyar, and V.Yu. Dubnitsky. The method of software reliability growth models choice using assumptions matrix. In *COMPSAC '02: Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*, pages 541–546, Washington, DC, USA, 2002. IEEE Computer Society.

M.R. Lyu, editor. *Handbook of Software Reliability Engineering.* McGraw-Hill and IEEE Computer Society, New York, 1996.

J.D. Musa. *Software Reliability Engineering: More Reliable Software Faster and Cheaper.* Author House, Bloomington, USA, 2nd edition, 2006.

H. Pham. *System Software Reliability.* Springer Series in Reliability Engineering. Springer, London, 2006.

S.E. Rigdon and A.P. Basu. *Statistical Methods for the Reliability of Repairable Systems.* Wiley, 2000.

L. Yin and K.S. Trivedi. Confidence interval estimation of NHPP-based software reliability models. In *Proc. 10th Int. Symp. Software Reliability Engineering (ISSRE 1999)*, pages 6–11, 1999.

J. Zhao and J. Wang. A new goodness-of-fit test based on the Laplace statistic for a large class of NHPP models. *Comm. Statist. Simulation Comput.*, 34(3):725–736, 2005.