Software Reliability and Security Assessment: Automation and Frameworks

Lance Fiondella¹ and Ying Shi²
University of Massachusetts Dartmouth¹
NASA Goddard Space Flight Center²
Software Failure and Reliability Assessment Tool (SFRAT)

• Implements software reliability growth models and relevant inferences
• **Known users** limited to DoD (NAVAIR), including supporting UARCs (JHU APL) and FFRDCs (MITRE, Aerospace Corporation) as well as major defense contractors (GD, Raytheon)
• Automation script and documentation published on SFRAT Github
Primary outputs and potential to communicate risk to management

Primary SFRAT outputs

- Trends in
  - Faults discovered
  - Time between failures
  - Failure intensity
- Reliability growth curve
- Predictions
  - Time to achieve specified reliability
  - Number of failures in specified time
  - Time to next $k$ failures

Potential benefits

- Visually and quantitatively identify progress toward software stability (less frequent/severe failure)
- Quantify probability of failure free operation for duration of mission
- Determine time required to achieve target reliability, time between failure, and failure intensity (corresponding schedule and cost risk)
SFRAT user modes

• Graphical user interface
  – Web and intranet

• Developer mode
  – Incorporate additional models

• Power user (present effort will support this class)
  – Streamline use for incorporation into internal software testing processes to encourage widespread application
  – Requires additional logic to remove human user from interface
Data requirements

• Failure Rate models
  – Inter-failure times - time between \((i - 1)^{st}\) and \(i^{th}\) failure, defined as \(t_i = (T_i - T_{i-1})\)
  – Failure times – vector of failure times,
    \[ T = < t_1, t_2, ..., t_n > \]

• Failure Counting models
  – Failure count data - length of the interval and number failures observed within it,
    \[ < T, K > = < (t_1, k_1), (t_2, k_2), ..., (t_n, k_n) > \]
Data requirements (2)

• The following will enable more accurate assessment and additional modeling
  – Time spent testing in each interval
  – Open and close times of defects
  – Severity
  – More detailed activity data in each interval
    • Execution time (hr), failure identification work (person hr), computer time failure identification (hr)
  – Cybersecurity
    • Penetration testing vs. vulnerabilities discovered
Summary of frequencies by DTT type and build found

Extracted defects and change requests from major and minor versions exhibiting a large number of events
Concatenated data

Concatenating data from successive minor versions without annotations lacks information about process
Tab 1 – After data upload

Cumulative failure data view
Cumulative failures

Plot enables comparison of data and model fits
SFRAT – File structure

- install_script.R
- server.R
- ui.R

utility
- Data
  - Data_Tools.R
- Metrics
  - GOF.R
- Plots
  - PlotModelResults.R
  - Plot_Raw_Data.R
  - Plot_Trend_Tests.R
- Prediction
  - Detailed_prediction.R
- tables
  - DataAndTrendTables.R
  - ModelResultTable.R
- RunModels.R

trend_tests
1. Laplace_trend_test.R
2. RAA.R

models
- GO
- DSS
- Wei
- JM
- GM

ISS
Time between failures

Times between failures should increase (indicates reliability growth)
Failure intensity should decrease (indicates reliability growth)
Reliability growth curve

Reliability growth vs. Cumulative Test Time

Reliability growth curve can estimate time to achieve target reliability
Time to achieve target reliability can help identify potential schedule overruns.

## Failure Predictions

<table>
<thead>
<tr>
<th>Model</th>
<th>Time to achieve $R = 0.9$ for mission of length 10</th>
<th>Expected # of failures for next 10 time units</th>
<th>Nth failure</th>
<th>Expected times to next 1 failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delayed S-Shape</td>
<td>342.43212</td>
<td>2.916884</td>
<td>1</td>
<td>3.349118</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td></td>
<td>3.57428</td>
<td>1</td>
<td>2.797767</td>
</tr>
<tr>
<td>Inflexion S-Shape</td>
<td>207.73900</td>
<td>0.639518</td>
<td>1</td>
<td>2.12015</td>
</tr>
</tbody>
</table>
## Model goodness of fit – AIC and PSSE

<table>
<thead>
<tr>
<th>Model</th>
<th>AIC</th>
<th>PSSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td>Delayed S-Shape</td>
<td>232.587035</td>
<td>290.840042</td>
</tr>
<tr>
<td>Goel-Okumoto</td>
<td>247.458558</td>
<td>341.069591</td>
</tr>
<tr>
<td>Inflexion S-Shape</td>
<td><strong>223.002</strong></td>
<td><strong>41.8493</strong></td>
</tr>
</tbody>
</table>

*Lower values preferred*
Power user mode

• Code can be tailored for internal use
  – Build into existing automated software testing procedures to provide near real-time feedback of reliability trends
  – Many industry standard programming languages can call R functions
    • Visual Basic, Java, C/C#/C++, and Fortran
    • Ensures tool will integrate smoothly
  – Python port should further enhance opportunities to incorporate into organizational processes
SFRAT Automatic Report Generation

.R file with SFRAT input specification

Markdown document to generate report
SFRAT Automatic Report Generation

Report can be Knit to pdf, Word, or HTML format
Sample output
Failure Count Predictions

Comparative performance based on 5-day ahead predictions
Potential future directions

• Develop simple models to characterize data open/close time distributions considering severity with practical goal of
  – Identifying test effectiveness in successive stages as measured by leakage/escape (covariates data may be helpful)
  – Finding and fixing with appropriate resource allocation
• Develop simple models to characterize the fault lifecycle including additional events between open and close
• Work with a program to identify decisions driving effective testing and reliable software
  – What happens between major (1.0), minor (1.1), a patches (1.0.1) that can help?
• Automated extraction from JIRA databases (completed by MITRE) and undergoing public release process
### Covariate data example

Could inform activity effectiveness and process improvement because parameters explicitly linked to activities

<table>
<thead>
<tr>
<th>week</th>
<th>Execution Time (hr)</th>
<th>Failure Identification Work (person hr)</th>
<th>Computer Time-Failure Ident. (hr)</th>
<th>Failure Identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0531</td>
<td>4</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.0619</td>
<td>20</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.1580</td>
<td>1</td>
<td>0.5</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0.0810</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1.0460</td>
<td>32</td>
<td>2.0</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>1.7500</td>
<td>32</td>
<td>5.0</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>2.9600</td>
<td>24</td>
<td>4.5</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>4.9700</td>
<td>24</td>
<td>2.5</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>0.4200</td>
<td>24</td>
<td>4.0</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>4.7000</td>
<td>30</td>
<td>2.0</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>0.9000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1.5000</td>
<td>8</td>
<td>4.0</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>2.0000</td>
<td>8</td>
<td>6.0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1.2000</td>
<td>12</td>
<td>4.0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1.2000</td>
<td>20</td>
<td>6.0</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>2.2000</td>
<td>32</td>
<td>10.0</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>7.6000</td>
<td>24</td>
<td>8.0</td>
<td>3</td>
</tr>
<tr>
<td>total</td>
<td>32.8000</td>
<td>296</td>
<td>60.0</td>
<td>54</td>
</tr>
</tbody>
</table>
Covariate model data fit

Version 2.1

Failed count

Time (t)

Observed failures
Model fit

Version 2.1
Acknowledgements

• This work was supported by the National Aeronautics and Space Administration (NASA) under Grant Number (#80NSSC18K0154) and NSF CAREER award (#1749635).