Abstract - Software reliability is the key concern of many users and developers of software. Although it is difficult to measure the reliability of software before its development is complete yet if data in terms of inter-failure time are available, predictions about its reliability can be made. It is with this objective that Jelinski – Moranda reliability model has been used in the present paper. However, there are other parameters like hazard rate, density function, etc. that affect the prediction. A real G.A. based evolutionary system using Simulated Annealing Methodology has, therefore, been developed to incorporate effectiveness in predicted reliability model. The model develops optimised simulation trajectory using Musa-data set. The application has been shown in this context.

Index Terms – Software Reliability, Jelinski-Moranda model, Hybrid Stochastic Search Techniques, Simulated Annealing.

I. INTRODUCTION

With the ever increasing role that software is playing in our systems, concern has steadily grown over the past few decades on ‘software quality’ characterised mainly by reliability besides other factors.

Software reliability is the probability of failure free operation of a computer program for a specified period of time in a specified environment. There are many reasons for software to fail but usually these are attributed to design problems resulting from new or changed requirements, revisions or corrections, etc. If existing problems, do not always create failures immediately then they are triggered only at certain states and inputs. Reliability is defined in terms of operational performance, something that one cannot measure before the product is finished. In order to provide reliability indicators before the system is complete a model is built of the factors that affect reliability and then its predictions are made based on one’s understanding of the system while it is under development ([1], [4]). A prediction system consists of a mathematical model together with a set of prediction procedures for determining unknown parameters and interpreting results ([1], [2], [3], [4]). A software reliability model specifies the general form of the dependence of the failure process on the principle factors that affect it, namely fault introduction, fault removal and the environment reliability. Such modelling has three broad stages viz., assessment, model development and measurement [6].

Different software reliability models can produce very different answers when called upon to predict future reliability in a reliability growth context ([8], [9]). Jelinski-Moranda reliability model has been concentrated upon for the prediction of the next time to failure in the present paper. To analyse these prediction, different measures have been used ([4], [5], [7]). For the sake of further convenience and effectiveness in
predictions, optimised simulation trajectory has been generated by evolutionary algorithm for predicted mean time to failure.

II. JELINSKI-MORANDA (J-M) MODEL

This model is credited with being the first reliability model. It belongs to a class of exponential order statistic model that assumes that fault detection and correction begins when a program contains N faults and all the faults have the same rate $\phi$.

A. Assumptions and Data Requirement:

The basic assumptions of the model are:

1. The rate of fault detection is proportional to the current fault content of the software.
2. The fault detection rate remains constant over the intervals between fault occurrence.
3. A fault is corrected instantaneously without introducing new faults into the software.
4. The software is operated in a similar manner as that in which reliability predictions are to be made.
5. Every fault has the same chance of being encountered within a severity class as any other fault in that class.
6. The failures, when the faults are detected, are independent.

Time taken between successive failures is the data required for the considered problem which has been taken from the Musa data set [4] as given in Appendix-I.

B. Model Form

From the overview of the model and the assumptions, one can determine that if the time-between-failure occurrences are $x_i = t_i - t_{i-1}$, $i = 1, \ldots, n$, then the $x_i$’s are independent exponentially distributed random variables with mean. Let $f(t_i)$ be probability density function for particular time $t_i$ such that –

$$f(x_i/t_{i-1}) = \phi[N - (i - 1)]exp(-\phi[N - (i - 1)]x_i)$$

and cumulative density function be

$$F_i(t_i) = 1 - \exp\left(-\lambda_i t_i\right)$$

and $1/\phi[N - (i - 1)] = 1/\lambda_i$

where, $\lambda_i$ is hazard rate.

Since this exponential model belongs to the binominal type, using equations (2), (3) one gets –

$$\mu(t) = N(1 - \exp(-\phi t))$$

$$\lambda(t) = N\phi(\exp(-\phi t))$$

where $\mu(t)$ is mean value function and $\lambda(t)$ is failure density function.

It is clearly a finite failure type model as –

$$\lim_{t \to \infty} \mu(t) = \lim_{t \to \infty} N(1 - \exp(-\phi t)) = N \ldots (6)$$

C. Model Estimation and Reliability Prediction

The maximal likelihood estimates, MLE’s, calculated from the joint density of the $x_i$’s, are the solutions to the following equations:

$$\hat{\phi} = \frac{n}{\hat{N}\left[\sum x_i\right] - \sum (i - 1)x_i} \ldots (7)$$

$$\hat{N} - (i - 1) = \frac{n}{\hat{\phi} - \frac{1}{\hat{N}}\left[\sum x_i\right]\sum (i - 1)x_i} \ldots (8)$$

A software program has been developed in C on Pentium III for the solution of equation (8). The solution so obtained is put in equation (7) for finding maximum likelihood estimates (MLE’s) $\hat{N}$ and $\hat{\phi}$.

Various reliability measures can then be obtained by replacing the quantities $\hat{N}$ and $\hat{\phi}$ in the reliability function of interest. As an example, the mean time to failure (MTTF) for the (n+1)th fault has its MLE as:
\[ \text{MTTF} = \frac{1}{\phi(N - n)} \]  \hspace{1cm} \text{(9)}

### D. Algorithm

The algorithm for solving equation (8) is given below:

1. \( \text{min} = 0.1 \)
2. For \( N=3 \) to \( N=50 \)
3. begin
4. \( \text{pl} = f(N) \)
5. if (\( p / < \text{min} \))
6. begin
7. \( \text{min} = p1 \)
8. print \( \text{min}, N \)
9. end
10. end

Musa data set has been used in the above program. The value of \( N \) is initialized. The algorithm gives value of \( N \) for which \( f(N) \) is minimum – where

\[
f(N) = \sum_{i=1}^{n} \frac{1}{N-(i-1)} - \frac{n}{N} \left( \frac{1}{\sum_{i=1}^{n} x_i} \right) \sum_{i=1}^{n} (i-1)x_i \]  \hspace{1cm} \text{(10)}

### E. Predictive Accuracy

The concept of Kolmogorov – Smirnov distance ([1], [4]) has been used to determine and analyse the accuracy of prediction system. For this purpose U-plot (Fig.1) and Y-plot (Fig.2) have been obtained as a measure of pessimistic or optimistic prediction as regards to its level of closeness with true distribution.

Further frequential likelihood function and Braun statistic concept has also been utilised to deal with the noisiness in system predictions. MTTF is therefore introduced for such analysis. Braun statistic is a measure of variability and gives the quantitative inference that how noisy a model is. Frequentional likelihood function is a technique particularly using narrow windows to detect changes in preferences and MTTF also helps in analysing the noisiness of prediction system by showing a series of fluctuations in its MTTF plot as depicted in Figure 3.

### III. Optimised Simulation Trajectory

It is prudent to make use of multiple statistical model to predict reliability quality of any predictive
environment as in the case of software reliability J-M model.

It would indeed be the most convenient for predicting reliability behaviour to generate optimised simulation in the predictive space defined by J-M model by applying Hybrid Stochastic Search Technique based on Simulated Annealing in G.A. which have proved to be effective as a solution tool for complex optimised problems in number of fields.

A. **Problem Solving**

Objective function:

Minimise $\text{MTTF} = K \frac{N - i + 1}{\lambda_i (N - n)}$

where, $K$ is the smoothing factor to bring unbiasedness and non-noisiness in the prediction accuracy.

Subject to

1. $0.002 \leq \lambda_i \leq 0.007$
2. $35 \leq n \leq 150$
3. $1 \leq i \leq 100$
4. $1 \leq N \leq 200$
5. $90 \leq K \leq 100$
6. $N \geq n$
7. $i \geq 1$

B. **Hybrid Stochastic Search Technique (HSST)**

HSST is a hybrid approach, which incorporates Simulated Annealing in the selection process of GA. Thus, HSST provides the advantages of both GA and SA. The HSST algorithm can be expressed concisely in the form of a pseudo-code as given below: ([10], [11], [12], [13])

**PSEUDO-CODE**

Initialise

(i) Set initial temperatures

(ii) Randomly select $N$ parent strings

(iii) Number of children to be generated by each parent

1. For each parent $i$, generate $m(i)$ children using crossover.
2. Perform mutation with a probability $pm$
3. Find the best child for each parent based on the fitness and constraints.

4. Select the best child as the parent for the next generation.
5. Repeat Step 7 to Step 10 for each family
6. Count=0
7. Repeat Step 9 for each child; Goto Step 10
8. Increase count.
9. Acceptance number of the family is equal to count $(A)$
10. Sum up the acceptance numbers of all the families $(S)$
11. For each family $i$, calculate the number of children to be generated in the next generation according to the following formula: $m(i) = (T \times A) / S$

$T =$ Total number of children generated by all the families.
12. Decrease the temperature
13. Repeat Step 2 to Step 13 until a certain number of iterations has been reached

The optimum solution has been shown in Table 1.

<table>
<thead>
<tr>
<th>N</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTTF (Model)</td>
<td>200</td>
<td>350</td>
<td>300</td>
<td>500</td>
<td>550</td>
<td>1100</td>
</tr>
<tr>
<td>MTTF (HSST)</td>
<td>214.2</td>
<td>256.5</td>
<td>321.3</td>
<td>428.4</td>
<td>642.6</td>
<td>1285</td>
</tr>
</tbody>
</table>

IV. **CONCLUSION**

Brawn Statistic, Prequential likelihood function, U-plot and Y-plot have been used to estimate the biasedness and noisiness of the software reliability as a measure of its quality. Kolomgrove-Simrnov distance has been used wherever necessary.

It has been observed that Kolmogov - Simrnov distance of Y-plot is less than of U-plot meaning thereby that the predictions in Y-plot are closer to true reliability for the adopted data set as shown in Figures (1) and (2).

The analysis given by MTTF for J-M Model infers that initially it is less
noisy but later becomes significantly noisy as depicted in Figure 3.

GA based Hybrid Stochastic Search Technique, has turned out to be good tool for optimised simulated trajectory for variables which are important performance indicators like $\lambda$, $\phi$ etc. to predict quality of reliability of the predicted software failure. The simulated trajectory is also shown in Figure (3).

The MTTF values have been found to be much closer for predictions which are less noisy. However, it can be inferred that more experimental runs would try to make better smoothing even at noisier ranges by varying the bounds of correction factor in the proposed Jelinski – Moranda model.

Such experiments will be useful for better reliability monitoring of softwares.

References

Appendix-I

Musa Data Set

Execution times in seconds between successive failures (left to right)

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>30</th>
<th>113</th>
<th>81</th>
<th>115</th>
<th>9</th>
<th>2</th>
<th>91</th>
<th>112</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>138</td>
<td>50</td>
<td>77</td>
<td>24</td>
<td>108</td>
<td>88</td>
<td>670</td>
<td>120</td>
<td>26</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>325</td>
<td>55</td>
<td>242</td>
<td>68</td>
<td>422</td>
<td>180</td>
<td>10</td>
<td>1146</td>
<td>600</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>227</td>
<td>65</td>
<td>176</td>
<td>58</td>
<td>457</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>263</td>
<td>452</td>
<td>255</td>
<td>197</td>
<td>193</td>
<td>6</td>
<td>79</td>
<td>816</td>
<td>1351</td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>21</td>
<td>233</td>
<td>134</td>
<td>357</td>
<td>193</td>
<td>236</td>
<td>31</td>
<td>369</td>
<td>748</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>232</td>
<td>330</td>
<td>365</td>
<td>1222</td>
<td>543</td>
<td>10</td>
<td>16</td>
<td>429</td>
<td>379</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>129</td>
<td>810</td>
<td>290</td>
<td>300</td>
<td>529</td>
<td>281</td>
<td>160</td>
<td>828</td>
<td>1011</td>
<td></td>
</tr>
<tr>
<td>445</td>
<td>296</td>
<td>1755</td>
<td>1064</td>
<td>1783</td>
<td>860</td>
<td>983</td>
<td>707</td>
<td>33</td>
<td>868</td>
<td></td>
</tr>
<tr>
<td>724</td>
<td>2323</td>
<td>2930</td>
<td>1461</td>
<td>843</td>
<td>12</td>
<td>261</td>
<td>1800</td>
<td>865</td>
<td>1435</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>143</td>
<td>108</td>
<td>0</td>
<td>3110</td>
<td>1247</td>
<td>943</td>
<td>700</td>
<td>875</td>
<td>245</td>
<td></td>
</tr>
<tr>
<td>729</td>
<td>1897</td>
<td>447</td>
<td>386</td>
<td>446</td>
<td>122</td>
<td>990</td>
<td>948</td>
<td>1082</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>482</td>
<td>5509</td>
<td>100</td>
<td>10</td>
<td>1071</td>
<td>371</td>
<td>790</td>
<td>6150</td>
<td>3321</td>
<td></td>
</tr>
<tr>
<td>1045</td>
<td>648</td>
<td>5485</td>
<td>1160</td>
<td>1864</td>
<td>4116</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>