CHAPTER 3

PROPOSED WORK

There are different software reliability models. First of all I have studied the software reliability models thoroughly. The models are described below

3.1.1 Jelinski Moranda Model:
The Jelinski Moranda model is the earliest models in software reliability. It is given in 1972. It is time between failure models. It assumes N software faults at the start of testing, failure occur purely at random and all the faults contribute equally to cause a failure during testing.

It also assumes the fix time is negligible and the fix for each failure is perfect. The software product’s failure rate improves by the same amount at each fix. The hazard function at time t is given

\[ Z(t) = [N-(i-1)] \]

Where N is the number of software defects at the beginning i is probability constant

3.1.2 Littlewoods Model:
It is similar to Jelinski Moranda Model but it assumes that different faults have different size. Large sized defects tend to detected and fixed earlier. As the number of errors is driven down with the progress in the test so is the average error size causing a law of diminishing return in debugging.
3.1.3 Goel Okumoto Imperfect Model:
The J-M model assumes that the fix is negligible and that the fix for each failure is perfect. It assumes perfect debugging. In practice this is not always case. In the process of fixing a defect new defects may be injected. This model proposed imperfect debugging model to overcome the limitation of the assumption. In this model the hazard function during the interval between the \((i-1)\) st and the failure is given

\[ Z(t_i) = [N-p(i-1)] \lambda \]

Where \(N\) is number of faults at start of testing
\(P\) is probability of imperfect debugging
\(\lambda\) failure rate per fault

3.1.4 Goel Okumoto Non Homogeneous Poisson Process Model:
The NHHP model is given in 1979. It is concerned with modelling the number of failures observed in given testing intervals. It defines the cumulative number of failure observed at time \(t\). \(N(t)\) can be modelled as a non homogenous Poisson process with the time dependent failure rate.

3.1.5 The Delayed S and Inflection S Models:
With the help of defect removal process Yamada said that a testing process consists of not only a defect detection process but also defect isolation process because of the time needed for failure analysis significant delay can occur between time of first failure observation and the time of reporting. They offer the delayed S-Shaped reliability growth model for such a process in which the observed growth curve of cumulative number of detected defects is S-shaped. The model is based on the no homogeneous poison process but with a different mean value function to reflect the delay in failure reporting.
\[ m(t) = k[1 - (1 + \lambda t)e^{-\lambda t}] \]

Where \( t \) is the time, \( \lambda \) is the error detection rate, and \( k \) is the total number of defects or total cumulative defect rate.

**Working:** There are more than 70 software reliability models. For finding the reliability of the software, we cannot able to find that which model should be used or finding the mean time to failure and failure rate we did not able to know that which model should be used.

The software reliability model can apply on different life cycle phases. The software Development life Cycle phases are:

1. Requirement Phase
2. Designing Phase
3. Implementation Phase
4. Testing Phase
5. Validation and Maintenance Phase

The above are software development life cycle phases. Then as we know there are more than 70 software reliability life cycle models. We don’t know which Software Reliability model apply on which phase of software development life cycle because we can use many models on one phase. We cannot able to apply the entire software reliability model on one phase because it requires more cost, more time consuming and large number of efforts are required.

From practical use of models for some models the cost of gathering data is too expensive. Some models are not understandable and some models don’t work when examined. Due to this we cannot use all the software reliability models. In the software development life cycle we cannot find the exact software reliability
by using one model. We have to apply different models on all the phases of software development.

The below is the diagram which represents that in the software development life cycle phases in which phase which software reliability models we can apply.

The above diagram represents the models used in SDLC. Now I have designed an algorithm which helps to find that which algorithm should be used.
Criteria for software reliability model selection

1) **Life Cycle Phase**: Different software reliability models are used in different phases. So it is difficult to find which model should be used in which phase.

Example: Input domain based model should be only used in the validation phase. We cannot use this model earlier.

2) **Output desired by the user**: In this we select only that model which provides the output desired by the user.

Example: If the mean time to failure required by the user we cannot use Goel Okumutu non homogenous model.

3) **Input required by the model**: If the input required the model is not available we cannot use that model.

Example: Software reliability growth model require mean time to failure if this information is not available then we cannot use SRGM.

4) **Trend exhibited by the data**: In the curve of collected failure data is matched with the curve of the then we use that model.

Example: If there is increase and decrease in the curve then we use S-shaped model.

5) **Validation of assumptions according to data**: In this the models with the high validation of assumption are selected.

Example: In the exponential models there is an assumption that the testing effort is homogeneous during testing phase. If testing effort is non-homogeneous then exponential models do not give good results.

6) **Nature of Project**: It includes that whether the application is terminating or non-terminating. It includes the size of the project.