CHAPTER 4
FORMULATION OF THE PROBLEM

4.1 Problems and Solutions:

Problem: - Sometimes it is difficult to find the nature of project or we have to find the reliability of whole software that is not for the particular phase.

Solution: - For calculating the reliability of different models we use different techniques. The

Maximum Likelihood Technique: One of the techniques for estimating parameters of models is the Maximum Likelihood Estimation technique or MLE. MLE has several desirable properties including that it is consistent, it is efficient, and it is asymptotically normal.

We will use MLE to estimate the parameters of each model. Maximum Likelihood starts with the likelihood function, $L(\beta, Y_D)$, where $\beta$ is a set of parameters, and $Y_D$ is a set of observed data. If the value of, $L(\beta, Y_D)$, is small, it means that for this set of parameters, it would be rare to see the observed data $Y_D$. Conversely if the value of, $L(\beta, Y_D)$, is large, it means that the data is likely to have come from a model with the parameters $\beta$. We then attempt to maximize $L(\beta, Y_D)$. To maximize the likelihood function, we solve the set of simultaneous equations that are comprised of the partial derivatives $L(\beta, Y_D)$, with respect to each parameter. These partial derivatives are often hard to work with, so the natural log of the likelihood function is often used. This function is called the log-likelihood function. Even using the simpler log likelihood function, the set of
simultaneous equations is very complex, and an iterative numerical solution is required.

### 4.2 Model Implementation:

For measuring the software reliability there are many models. We cannot use all the models at same time. It means that like or the design phase we can use early prediction models or architecture based models. Now these models have further sub models which is more than 20. We cannot use all these models for the prediction of software reliability.

Now my work is to design an algorithm to select the software reliability models. In this algorithm I have given standard weights and relevant weights. On the basis of weights the model which is having highest weight should be selected.

At last the algorithm is validated by taking the example which shows that the model which is having highest weight is selected.

**Discussions:** Following discussions are made after the result was analysed:-

#### 4.2.1 How this algorithm assigns weight:

The weights are assigned on the basis of

- Input desired by Model
- Nature of project
- Output desired by user
- Testing
- Validation
4.2.2 Precautions which should be taken:

For choosing the best model to find the software reliability, first of all carefully find the SDLC phase. Then the weights should be assigned properly. If the weights are assigned incorrectly the wrong result should be found. Find the nature of the project, input desired by the model, output desired by the user, testing process properly. For example if there is three inputs available out of four and we are assigning the weight of two inputs then the wrong result should be found.

- How can we choose model: -

There are many software reliability models we cannot use all the models. The model is chosen on the basis of SDLC phases. For example if SDLC phase is Implementation Phase we can use early prediction Model and architecture Based model.
4.3 EXISTING SOFTWARE RELIABILITY MODEL

The author of [48] proposed a software reliability growth model on the basis of the existing model like logistic growth curve model etc. The model is designed on the basis of the tangential function. The tangential model must be drawn in the positive axis. It varies from 0 to infinity similar to the software reliability. The software reliability is inversely proportional to the fault detection as the no of fault detection decrease the reliability increases. The zero fault detection means the infinite reliability and the zero software reliability means the infinite faults. The proposed model suits the behavior of the software reliability so fits to the software reliability.

In the beginning of testing, there is exponential number of faults in the software code. The number of faults is unknown but they are fixed in number. All faults are of same type. Each fault can be detected independent of each other. The remaining number of fault and the remaining time is useful to determine the other parameters. The probability of occurring of each fault is same. Each fault occurred can be removed instantaneously. The mean value function can be given as

\[ m(t) = \frac{fv - \exp((-\phi t))}{\{1 + \exp((-\phi t))\} > 0?fv:0 \]  \hspace{1cm} (1)

The failure intensity can be expressed as

\[ \lambda(t) = \frac{dm(t)}{d(t)} \]  \hspace{1cm} (2)

According to the failure intensity of the software at time t is proportional to the expected number of faults remaining in the software.