Conclusion

Software Reliability Engineering is an emerging discipline whose importance cannot be undermined. It is receiving unprecedented attention from researchers. In this thesis, it was endeavoured to develop more practical resource allocation approach catering to different fault removal models under dynamic environment. The thesis is divided into five chapters. The first chapter briefly discusses some theoretical concepts and literature related to the work presented. A brief description of the work presented in rest of the chapters and scope of future research is discussed.

In section 1 of chapter 2, using optimal control theory, a systematic technique for the software project managers to allocate the resources for detection and correction, when a company has limited budget has been developed. It is observed that initially the path of detection effort increases but after a certain point of time, it attains a saturation level. Also, when most of the simple faults have been debugged, there will remain some hard and complex faults though detection of these faults will no longer remain difficult due to learning curve phenomenon. Due to this, the path of correction effort initially decreases and then attains a saturation point. GA is applied successfully to obtain optimum value of detection and correction effort when the company has limited budget. In addition to this, a trade-off between reliability and release time of a software project is examined. Using sensitivity analysis, we have observed the pattern of correction cost and variation in optimal allocation with respect to the parameters. We observed that, in a cut throat market, managers are better off underestimating the cost of correction left in the system, resulting in a choice of project duration that is below the desired level. In section 2, optimal policy has been developed for proposed resource allocation model for multiple releases under dynamic environment. Using GA, optimum value of testing efforts for each release is obtained for the particular case considered. From the comparison analysis, it is concluded that testing efforts and faults detected for each release of our proposed model & real data set are closely related to each other. Graphical illustrations are also presented displaying the above relationships.

In chapter 3, optimal resource allocation model considering different budgetary constraint for fault detection and fault correction process has been developed. Two cases are considered to obtain the optimal policy for detection and correction effort
using Optimal Control Theory. In first case, the cost for detection and cost for correction are considered as constants and it is observed that optimal policies for detection and correction effort are the combination of generalized bang-bang and singular controls. In second case, Pegels (1969) functional form has been incorporated in detection and correction cost. Simulations are carried out using different values of model parameters and graphs are plotted between faults detected and faults corrected for this case. These graphs shows that there is always a time-lag between detection and correction process. This is signified by the gap between the lines showing faults detected and faults corrected. Also, graphs of shadow cost of detection and shadow cost of correction are illustrated. In addition to this, based on the theoretical study, it is concluded that the optimum release time is the time where the total cost of correction coincides with the total cost of detection, maintaining the strict reliability criteria. Else, the firm should wait for an ideal time.

In section 1 of chapter 4, the model proposed integrate the effect of both testing time and resources and considered the testing phase to be a two-stage process of fault detection and fault correction in perfect debugging environment. The fault detection is taken as an exponential distribution function and fault correction is taken as Gamma distribution function. It also takes into consideration the left over faults of the previous release along with new fault content. Thus, a two-dimensional multi-release framework for two-stage testing process is developed. The parameters estimated are further validated on real data sets and the goodness fit is also calculated. In section 2 of chapter 4, testing time and effort dependent multi release modeling under the effect of two types of imperfect debugging has been developed. In the proposed modeling, we have considered testing phase to be two-stage fault detection and fault correction process. Here, SRGMs are proposed from existing SRGMs using well-known Cobb Douglas function. The left over faults in the previous release are also incorporated in addition to the new fault content. Validity of the proposed SRGMs has been tested on Tandem Computer data set. In addition, the comparison analysis of the proposed SRGMs is made by five common criteria namely, Bias, MSE, $R^2$, Variation and RMSPE. The graphs are also illustrated to depict the good fit results of both SRGMs for all releases.
In section 1 of chapter 5, Shannon entropy, Renyi entropy and Tsallis entropy have been calculated using the observed bugs for each subcomponents of Mozilla open source software. Renyi and Tsallis entropy for five different values of $\alpha$ are considered i.e. 0.1, 0.3, 0.5, 0.7 and 0.9. It is concluded that Shannon entropy lies between 2 and 4. Renyi and Tsallis entropy decreases as $\alpha$ increase. It is observed that the predicted bugs evaluated are closely related to the observed bugs for each time period. $R^2$ is maximum in case of Shannon entropy amongst the entropy measures considered. For Renyi and Tsallis entropy as $\alpha$ increases from 0.1 to 0.9; $R^2$ also increases i.e. $R^2$ is maximum for $\alpha = 0.9$. ANOVA and Tukey test results are also depicted. In section 2 of chapter 5, an approach is proposed to predict time for each release of another open source software- Bugzilla with the help of regression analysis. Regression analysis has been applied in two steps, in first step with the help of simple linear regression, bugs are predicted for each release and in second step using these predicted bugs and entropy measure, predicted time of each release has been calculated. For each entropy measure viz. Renyi entropy and Havrda-Charvat entropy; four parameter values are considered i.e. 0.1, 0.3, 0.5 and 0.7. Comparison between the observed and predicted value is depicted using the goodness of fit curve. From the graph it is observed that there is no significant difference between the observed and predicted value for all the releases considered. Amongst all the entropy measures considered for different parameter values, $R^2 = 0.897$ is maximum for Renyi entropy when $\alpha = 0.1$ and $p$ value in this case is 0.000. Therefore, in this case, $R^2$ is statistically significant.

**Future Scope**

1. In this work, we have only analyzed the effect of deterministic models for detection and correction process. Stochastic models for fault detection and fault correction process can be taken into consideration for future work.

2. Incorporation of the effect of perfect debugging and imperfect debugging with error generation in a multi release modeling using two stage fault detection and fault correction process is seen in our work. We can extend this work by
considering the impact of testing coverage and number of test cases executed in fault correction process.

3. For future research, cost function can also be incorporated in a multi release modeling using two stage fault detection and fault correction process.

4. Mozilla and Bugzilla are only taken into consideration for prediction modeling of open source software systems. Other open source software systems can be studied for future work. In addition to this prediction modeling may also be carried out in case of closed source software systems.

5. For prediction modeling, our study is confined to few entropy measures viz. Shannon entropy, Renyi entropy, Tsallis entropy and Havrda-Charvat entropy and the parameter values for these measures are limited to few values. This work may be extended by considering other entropy measures with different parameter values for prediction of bugs and release time of the software system.