A Survival Study on Software Failure Prediction Using Adaptive Dimensional Gene Model

V. Sangeetha
Department of Computer Science
Periyar University College of Arts & Science
Pappireddipatti, Dharmapuri, Tamilnadu, India

P. Saravanan
Department of Computer Science
Dharmapuri Arts College, Dharmapuri
Tamilnadu, India

***Abstract*- Software reliability is a probability of software system to function under operating conditions over period of time. Software reliability concentrates on prediction of residual faults, estimation of failure intensity, reliability and cost. Software faults causes due to the software failures generated through the developmental process of software. Software system failures have large drawbacks on time and resources for recovery. A software system failure is predictable and large attention is given to software system failure prediction.However, the probability of failure estimation remained unsolved. The research work is carried out to perform the fast prediction of software failures through Genetic-based Bayesian model for improving the true positive rate.**

Keywords- Software reliability, Software faults, failure prediction, software system, Genetic-based Bayesian model

1. INTRODUCTION

Failure analysis techniques classify the dependability actions of operational systems in industrial sectors like aerospace and automotive. Failure techniques identifies the system failure modes, failure causes, occurrence prevention and increases the future system dependability. Failure analysis gathers the event logs and system-generated files during operations. Software system failures have large drawbacks on time and resources for recovery. An efficient forecasting for software system failure is predictable and increasing the attention to the software system failure prediction. The technologies are used in software systems from GUIs to distributed one. Network communication frameworks are highly complex whose behavior is determined by incoming events called as Event-Driven Software (EDS). The software errors solving in EDS is the failure because of the complex network structure.

This paper is organized as follows: Section 2 discusses reviews on software fault prediction techniques, Section 3 describes the existing software fault prediction techniques in gene model, Section 4 identifies the possible comparison between them, Section 5 explains the limitations and Section 6 concludes the paper, key areas of research is given as to perform the fast prediction of the software failures through Genetic-based Bayesian model for improving the true positive rate.

2. LITERATURE SURVEY

A Rule Based Logging (RBL) approach (Marcello Cinque D. C., 2012) is used for analyzing the software failures to improve recall and precision rate. The scheme leverages artifacts created at system design time and designed rule set for formalizing the provision of logging instructions inside the source code. RBL approach improved detection capability of log and the analysis was carried out with recall and precision. But the approach was proved to be inefficient when service was invoked with bad parameters. Recovery from software failures caused by Mandel bugs and recovery process was presented (Michael Grottke, 2015). An analytical model fixes software failures by Mandel bugs depending on systematic mean time recovery with numerical and parametric sensitivity analysis. Though analysis was efficient with mean time to recovery, failure causes remained unaddressed.

Linguistic values of software metrics in fuzzy inference system (Subhashis Chatterjee, 2016) was used for fault prediction in software through phase analysis. A fuzzy rule based algorithm identifies both the input

and output metrics. Characterizing Monitoring techniques depending on Fault Injection (CM-FI) method (Marcello Cinque, 2016)calculates the precision and recall. It also evaluated the dissimilarity of data generated upon failures. Empirical Bayes Software Reliability (EBSR) model (Barraza, 2015) was used to calculate the mean time failures in software testing. EBSR model calculated failure frequency up to a given time interval towards estimating the probability of failure appearance during the next time interval. The estimate was made using mixed Poisson model and Probability Density Function to evaluate the reliability rate. However, the estimate of the probability of failure remained unsolved.

Relative Defect Proneness (Mark D. Syer, 2015) was analyzed by survival analysis modeling effect to study the influence of size defects for guaranteeing the size-defect relationship. The impact of modeling effect with discrete time scale measured the observations at particular points in particular time interval. It also guarantees the size-defect relationship where defect density peaks are to be addressed. An integrated approach called Random Indexing (RI) and Support Vector Machines (SVMs) (Ilenia Fronza, 2013)predicted the software failures depending on log files that lead to classification of non-failures. The RI model symbolizes the sequences where every operation was classified with its context. SVMs were applied to resultant RI value for classifying the failures or non-failures. But, the true positive rate is not at required level.

Event Driven Web Application Testing (EDWAT) process (Elahe Habibi, 2015) addresses the six stage process to overcome Event Driven Software (EDS) testing challenges. EDWAT applied Structural Division method to break the application into different sections. Followed by this, graphs were created in the second stage. From the resulting graphs, the mutated graphs were obtained. With mutated graphs, the coverage criteria were chosen to create the test paths for longer sequences creation by merging event sequences.

Model Based Mutation Testing (MBMT) (Fevzi Belli, 2015) fixed the single events issues and operated on sequences of events through changing the morphology. K-Sequent Right Regular Grammar was introduced for efficient representation of relation between events sequences and single events. Though the cost effectiveness of model was proved, cost factor with software failure prediction was not addressed. A new failure prediction method is introduced (Roberto Baldoni, 2012) for mission critical distributed systems with distinctive features. Complex Event Processing (CEP) and Hidden Markov Models (HMM) are joined to examine the failures symptoms in form of anomalous conditions. But, the prediction accuracy is not at required level. A fracture principle is designed (Dae-Young Kim, 2016) with micro-mechanics void model to ductile fracture model of AZ31B magnesium alloy sheet. McClintock fracture model is a fracture principle for magnesium alloy sheet with shear deformation effects to original model. But, the reliability rate is not enhanced.

Least squares support vector machines (LSSVM) are mainly used (Changxing Zhu, 2015) for solving the complex nonlinear issues. A LSSVM-based rock failure principle mainly examines the deformation of circular tunnel lacking function form. However, the true positive rate is not higher. The fracture mechanics approach and intrinsic defect concept are combined (Ben Hassine, 2014) to increase the failure prediction rate. By Ethylene–Propylene–Diene Monomer (EPDM), the average molar mass of active chains is employed as the indicator of macromolecular network degradation. But the reliability is not improved.

A multi-step fuzzy bridged refinement domain adaptation algorithm (Vahid Behbood, 2015) uses the similarity ideas for changing the labels of target cases predicted by shift-unaware model. It improves the labels by the instances same as the target instance. The designed algorithm functions on data and reprocess the labels in shift-unaware prediction mode. The practical applicability of predictive models is used (Manjula Gandhi Selvaraj, 2015) and demonstrated the models in project to predict the system testing defects. The model helps to minimize the occurrence of residual defects. However, the software prediction time is not reduced.

Evidence relating to probability of failure is identified with evidence relating to the probability of fault-freeness in Bayesian framework (Lorenzo Strigini, 2013). The probability is used for increasing the reliability predictions in spite of the complexity of stating prior probability distributions. The machine learning techniques are used (Arunima Jaiswal, 2016) for software reliability prediction and compute the software reliability prediction depending on selected performance principle. The match distribution curve method (Ping Jiang,

2014) was introduced to calculate the failure probability in Weibull distribution. A method is introduced to calculate the failure probability for Weibull distribution through the concavity or convexity. The failure probability was computed using Bayesian approach. A new methodology (Houssany. S, 2012) designed calculates the real cache sensitivity and finds the exact failure rate. The methodology is used for observing the cache accesses and needs the microprocessor simulator. But, the precision is not increased.

Three improved weighted-combinations are weighted arithmetic, weighted geometric and weighted harmonic combinations. For addressing the proper weights for model combinations, enhanced genetic algorithms (EGAs) are designed (Chao-Jung Hsu, 2014) with many operators into weighted assignments. A software reliability model (Hellmich, 2016) includes the operational profile of software. An instantaneous software failure rate is based on number of remaining faults and state of software system in Markov process. Statistical inference is carried out through linear filtering by innovative method and martingale theory. However, the failure prediction is not carried out in efficient manner.

A technique (Julio Fernandez-Ceniceros, 2014) is designed for increasing the performance of soft computing meta models and predicts the failure of steel bolted links. The parameters of meta models are adjusted using optimization depending on genetic algorithms in training process. The method comprises the selection of relevant input features and minimizes the complexity. T-stub bolted connection authenticates the meta models with better prediction accuracy. A method predicts software failures in subsequent agile sprints and attained the analytical and statistical methods. Analytics-driven testing (Feras A. Batarseh, 2015) predicts errors and their locations. It is carried out by calculating MTBF for software components. Forecasting regression model is designed for calculating the software system failures. But, the probability of the software prediction is not enhanced.

3. SOFTWARE FAILURE PREDICTION TECHNIQUES

Software bug analysis has common theme in many IT project implementations. The key objective is to minimize the impact and costs of bugs in software development projects through avoiding the introduction of bugs, and by improving the bug removal in development stage. The bug analysis leads to the guidelines and practices ranging from test case for designing the root cause analysis, checklists, programming and debugging performances.

Software reliability models are stochastic processes that establish the metrics on the failure detection process like MTBF and MTTF or number of remaining failures at any time. The metrics estimate the software development and testing processes for allocating the testing resources or for forecasting the release time. The feature of failure detection process describes the decreasing rate as testing time progresses. Many software reliability models were discussed for increasing the software the reliability growth models.

3.1. Direct Monitoring Techniques in Software Systems

The target software system use monitoring techniques to assess. The System under Test (SUT) is used with fault load and workload. The fault load includes the software fault that denotes the common programming mistakes in real software systems. The workload is distinctive operational profile for SUT. The execution of SUT based on the fault load and workload enables the failures into SUT. It is essential to measure level of direct monitoring methods by SUT to report failure existence through the fault load. With this idea, Monitoring technique under test (MUT) is carried out. The precision and recall of MUT is calculated. MUT has ability for monitoring the data on failure existence and the dissimilarity of data are created across many failures.

Fault load has collection of software faults established in the source code of software systems. The faults belong to the well-consolidated orthogonal defect classification(ODC). Fault types are used and extended ODC classes for practical injection purposes. Every fault of fault load is inserted into SUT. SUT is implemented to trigger the fault and to increase the failure occurrence. The workload is SUT dependent and failed to vary across the faults emulate. A general classification classifies the failures induced in SUT. It is classified as crash, silent, erratic and no failure. In crash failure, unexpected termination reviews the software that is deal located by

operating system. In silent failure, SUT is awake and the output/functionality is not presented in expected timeout. The expected timeout is presented through fault-free runs of system. For Erratic failure, exceptional conditions and errors failed to result in crash or silent failures. No failure manifestation is monitored during the experiment where the injected fault is failed to activate. An experiment model is used to gather the data generated by MUTs. In every experiment, one fault belong to the fault load is inserted into SUT. SUT with workload and monitoring data by MUTs are gathered.

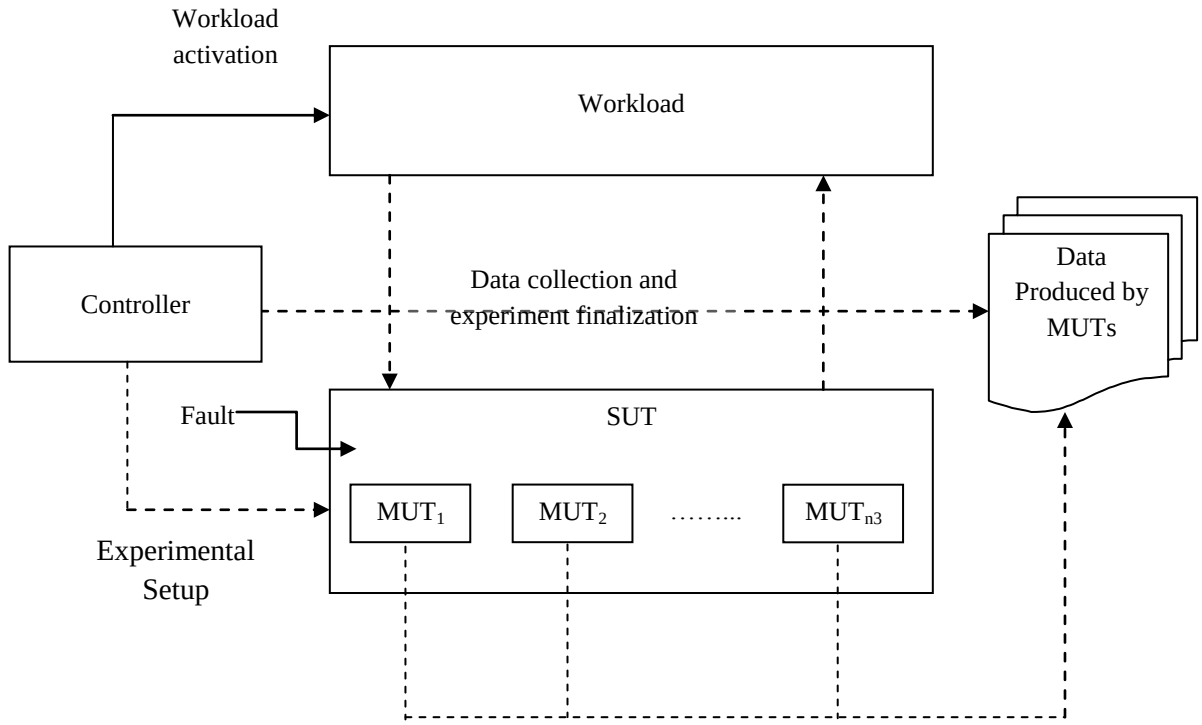

Figure 1. Assessment Approach

Figure 1 explains the step process used for all experiment. The execution of experiments is computerized and managed by controller program. The controller inserts the faults, stops SUT and reboots to guarantee the similar initial conditions for all experiments. In Experiment setup, one fault from the fault load is introduced in SUT. The patch file created by SAFE tool is controlled by controller to source code of SUT. The SUT gets compiled and initiated. During the workload activation, SUT is executed with workload. The workload uses SUT for fault triggering and activating the MUTs for many error conditions.

In data collection and experiment finalization, monitoring data created by MUT are saved in file in case of workload completion or predefined timeout expiration. After saving the files with generated data by MUTs for subsequent analysis, the SUT is returned, files are cleaned and the machines are rebooted before the next experiment. The controller finds occurrence of failure upon experiment completion. Controller examines the operating system-level data execution and workload-level data to establish failure type in adopted failure model. The output of experiment includes fault type induced in SUT, failure type and files with monitoring data created by every MUT.

3.2. Fuzzy Rule based Algorithm for estimation

Software metrics computes properties and requirement of software and software development process for attaining higher productivity and software quality. When the failure data is not present in software development, software metrics is an essential one in fault prediction. The linguistic values of software metrics are used as an input of fuzzy rule based fault prediction model. The software metrics are called as an input metrics. There are two kinds of metrics, namely process metrics and product metrics. Process metrics increases

the software development process and preservation. Product metrics increase the quality of different system component. The software metrics used in requirement phase are called as requirement metric.

Fuzzy inference system connects an input space to output space by fuzzy logic. Fuzzy inference engine includes the membership function, fuzzy Max–Min operator and 'if–then' rules. The output variable requires the defuzzification after processing the inference engine to attain the crisp set from fuzzy set. 'if–then' rule based model is used for fault prediction in early phase of software life cycle. The linguistic values of top ranked reliability related software metrics, metrics and target reliability are taken as input to predict total number of faults. The designed model comprises five parts, namely software metrics selection, fuzzy profile and membership function, inclusion of target reliability, fuzzy rule base development algorithm, fuzzy inference system process and total number of faults prediction in the software.

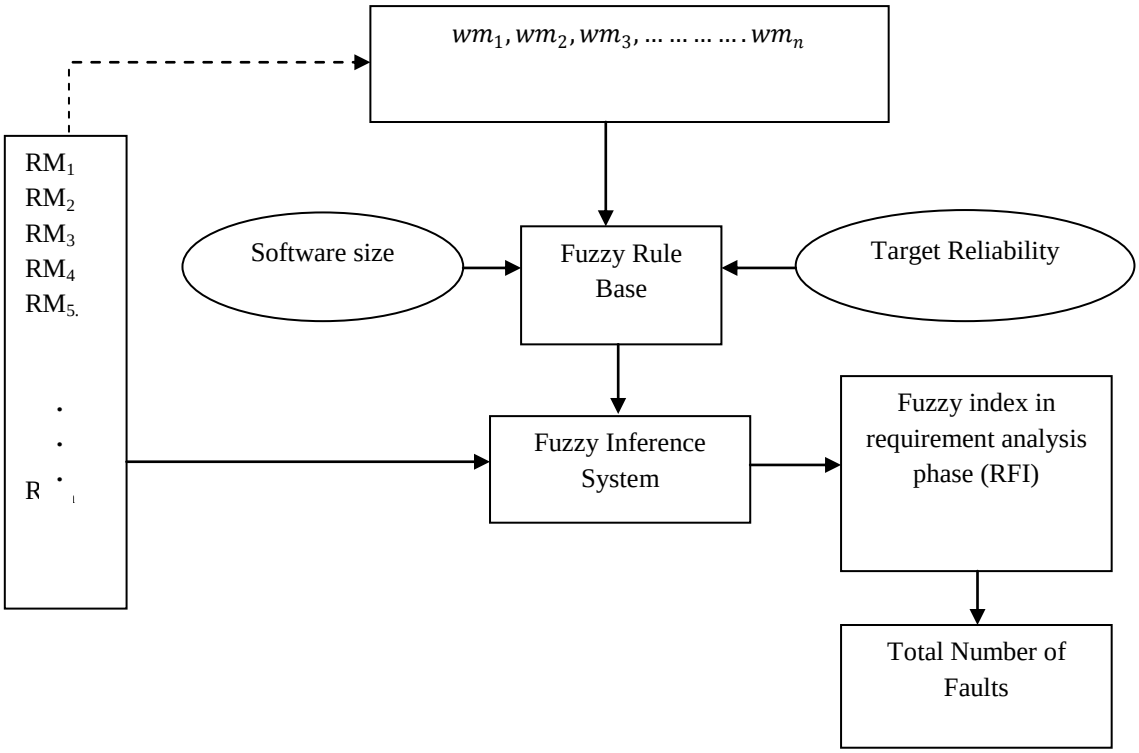

Figure 2. Fuzzy Rule Based Algorithm for Software Faults Estimation

Figure 2 explained the fuzzy rule based algorithm for software fault estimation. Software failure during the operation results in loss of economy or human lives. It is essential to fix target reliability for software in development process. The reliability is not same for software like real time control systems with high reliability than simple application software. The development cost is higher for attaining high target reliability. The complete set of input variables generates many fuzzy rules. Fuzzy inference process connects all fuzzy input sets to fuzzy output set as fuzzy 'Max–Min' operator. Fuzzy 'if–then' rule is used for the fuzzy inference process. Many faults in software are occurred in development phase. The faults are based on software metric condition at software development process. The condition of metrics is represented by linguistic value on ordinal scale.

3.3. Morphology Model for Event-Based Testing

Testing is a user-centric quality assurance method depending on test cases with test inputs and predictable test behaviors. A test uses the training of system under consideration (SUC) by test case. SUC exceeds the test value on test input and expected behavior is created. A set of test cases is termed as test set/suite. It is created and performed in target environment of SUC or an environment similar to the target environment. SUC is preferred to system under test (SUT) with both model and implementation. The model morphology is classified by length and contextual relation of event sequences. Through changing the sequence length, scalability of approach gets changed by generating equivalent model series from original one. The model utilization differs principally from the existing ones that create many kinds of models for many views.

4. PERFORMANCE ANALYSIS OF SOFTWARE FAILURE PREDICTION TECHNIQUES

In order to compare the software failure prediction using different techniques, number of test cases is taken to perform the experiment. Various parameters are used for software failure prediction.

4.1. Software Failure Identification Time

Software failure identification time is defined as time taken to identify the software failure. It is also defined as the difference of ending time and starting time for software failure identification. It is measured in terms of milliseconds. The software failure identification time is calculated by using equation 1. When the software failure identification time is lesser, the method is said to be efficient.

$$Software\ Failure\ Ientification\ Time = Ending\ time - Starting\ time\ of\ software\ fault\ identification \quad (1)$$

Table : 1 Tabulation of Software Failure Identification Time

Number of Test cases (Number)	Software Failure Identification Time (ms)		
	Direct Monitoring Techniques	Fuzzy Rule based Algorithm	Morphology Model
5	19	15	21
10	22	18	24
15	25	21	26
20	28	23	29
25	31	24	32
30	34	27	35
35	37	31	38

Table 1 explains the software failure identification time comparison for different number of test cases in the range of 5 to 35. The software failure identification time comparison takes place on existing Direct Monitoring techniques, Fuzzy Rule based Algorithm and Morphology Model. Figure 3 describes the software failure identification time comparison of existing techniques. As the number of test cases increases, software failure identification time gets increased automatically. The experiment shows that Fuzzy Rule based Algorithm consumes less software failure identification time than Direct Monitoring Techniques and Morphology Model. Research in Fuzzy Rule based Algorithm consumes 23.44% lesser software failure identification time when compared to Direct Monitoring Techniques and consumes 29.8% lesser software failure identification time when compared with Morphology Model.

4.1. True Positive Rate

True positive rate is defined as the ratio of number of test cases where the software faults are correctly predicted to the total number of test cases. It is measured in terms of percentage (%).The true positive rate is calculated by using equation 2. When the true positive rate is higher, the method is more efficient to be.

$$True\ Positive\ Rate = \frac{number\ \ of\ test\ cases\ where\ test\ cases\ correctly\ \ predicted}{Total\ \ number\ \ of\ test\ \ cases} \quad (2)$$

Table 2 explains the true positive rate comparison for different number of test cases in the range of 5 to 35. The true positive rate comparison takes place on existing Direct Monitoring techniques, Fuzzy Rule based Algorithm and Morphology Model.

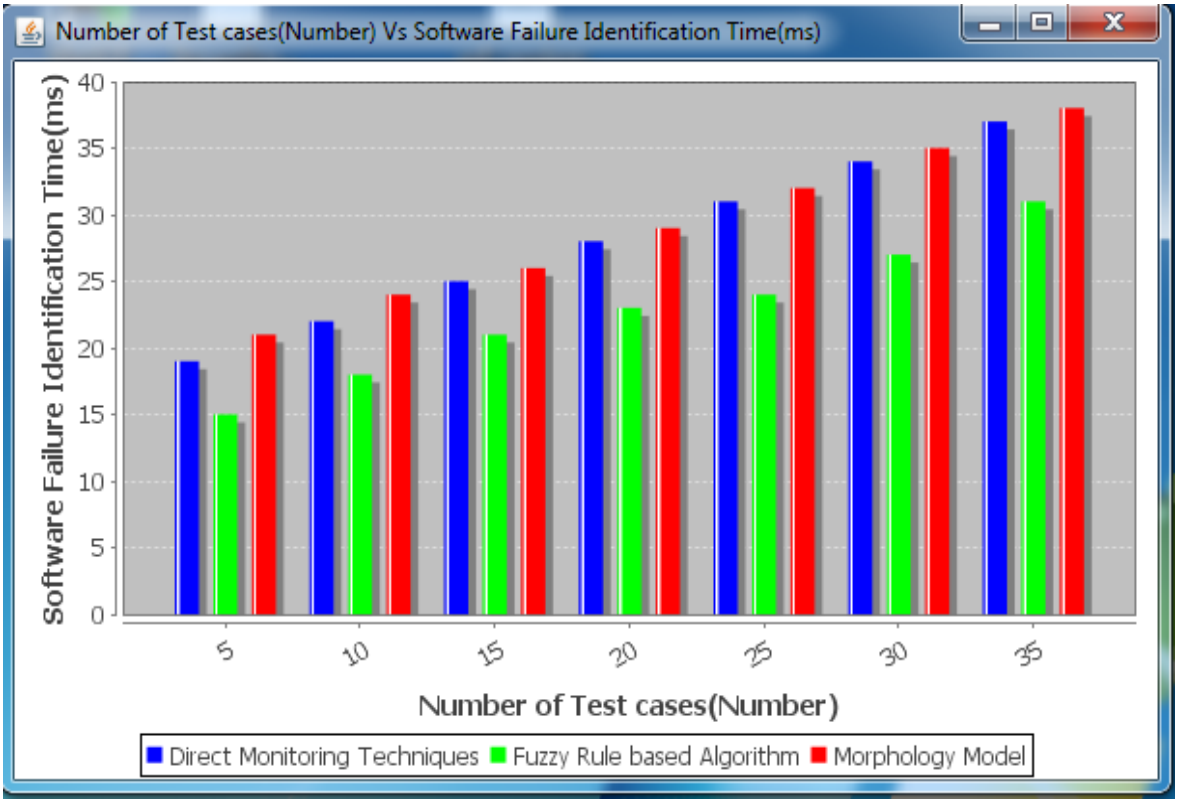

Figure 3. Measurement of Software Failure Identification Time

Table : 2 Tabulation of True Positive Rate

Number of Test cases (Number)	True Positive Rate (%)		
	Direct Monitoring Techniques	Fuzzy Rule based Algorithm	Morphology Model
5	79	65	74
10	81	68	76
15	84	71	78
20	87	74	82
25	89	77	85
30	91	80	87
35	93	82	90

Figure 4 describes the true positive rate comparison of existing techniques. As the number of test cases increases, true positive rate gets increased automatically. The experiment shows that Direct Monitoring Techniques increases the true positive rate when compared with Fuzzy Rule based Algorithm and Morphology Model. Research in Direct Monitoring Techniques has 14.5% higher true positive rate when compared to Fuzzy Rule based Algorithm and has 5.35% higher true positive rate when compared with Morphology Model.

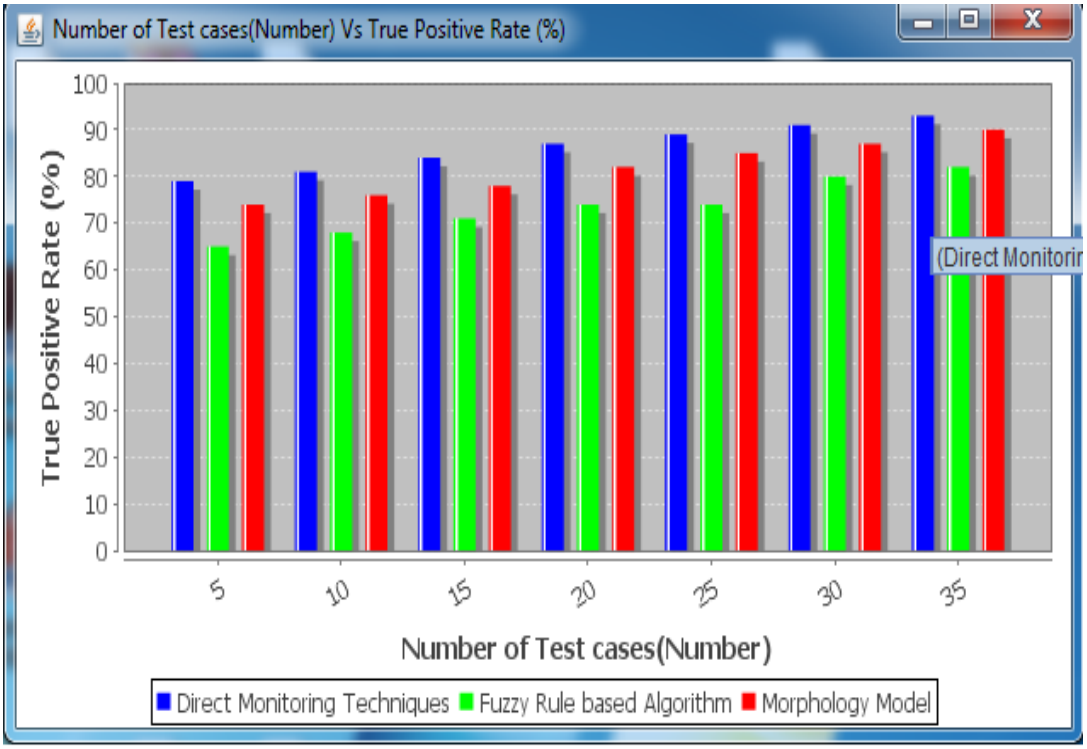

Figure 4. Measurement of True Positive Rate

4.2. Reliability Rate

Reliability rate of a software system is a measurement about how well it provides the services with the user's expectation. It is measured in terms of percentage (%). When many software failures occurred, the system is said to be less reliable.

Table : 3 Tabulation of Reliability Rate

Number of Test cases (Number)	Reliability rate (%)		
	Direct Monitoring Techniques	Fuzzy Rule based Algorithm	Morphology Model
5	65	71	84
10	67	74	86
15	69	77	89
20	72	79	91
25	75	82	93
30	78	85	95
35	81	88	96

Table 3 explains the reliability rate comparison for different number of test cases in the range of 5 to 35. The reliability rate comparison takes place on existing Fuzzy Rule based Algorithm, Direct Monitoring techniques and Morphology Model.

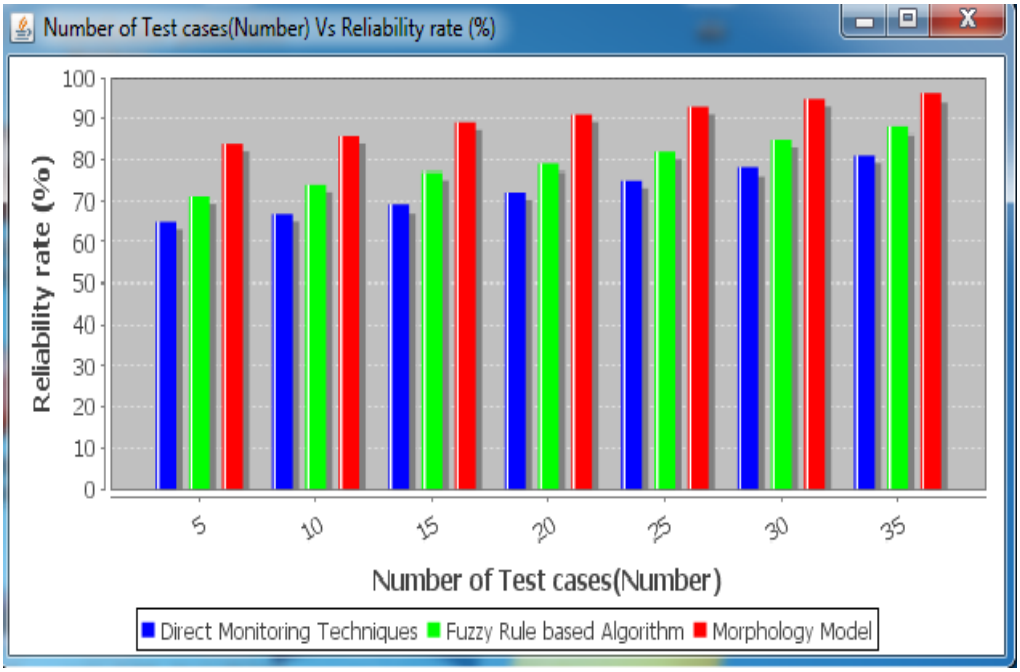

Figure 5. Measurement of Reliability Rate

Figure 5 describes the reliability rate comparison of existing techniques. As the number of test cases increases, reliability rate gets increased automatically. The experiment shows that Morphology Model increases the reliability rate when compared with Fuzzy Rule based Algorithm and Direct Monitoring Techniques. Research in Morphology Model has 20.1% higher reliability rate when compared to Fuzzy Rule based Algorithm and has 12.3% higher reliability rate when compared with Direct Monitoring Techniques.

5. DISCUSSION ON LIMITATION OF SOFTWARE FAILURE PREDICTION TECHNIQUES

Monitoring technique is used in software system based on fault injection approach through measuring accuracy and recall. The method aimed the failure analysis with reporting capability and many techniques by test. Monitoring techniques in two software systems comprises communication middleware and arrival manager. Direct monitoring techniques is used by event logging, assertion checking and source code instrumentation techniques. Though, indirect monitoring techniques failed to explore monitoring software system and increase software failure.

The fuzzy rule based inference system is designed with linguistic values of software metrics to forecast the total amount of faults in development phase. The value of metrics and target reliability is taken as input for fuzzy rule base model for the fault prediction. For optimizing the testing effort, cost and schedule required software development process. The value of software metrics is not taken for the reliability of fixed faults. Fuzzy inference system is complex to define the prediction of software faults. An event- based and model-based mutation testing approach changes the grammar model to morphologically create many models. Morphology model performs the mutants and multiple mutants. The different morphology models allow systematic generation of mutants and increases the set of fault models. The different hierarchy and communication modeling features of particular systems are not considered. Morphology model failed to use with higher semantic using generalization of grammar transformation.

5.1. Future Direction

The future direction of the software failure prediction is to perform the fast prediction of software failures through gene model with better true positive rate. Hybrid gene model can be used to further increase the performance of the failure prediction and reduce the software failure identification time. In addition, Bayesian

model can also be used for calculating the probability of failure occurrence in the software in effective manner. In order to increase the prediction rate, the support vector machine (SVM) classifier is to be used which increases the failure prediction rate reliability rate in the software.

6. CONCLUSION

In this paper**,** survival study is taken for software failure prediction through gene model. Indirect monitoring techniques failed to explore monitoring software system and increase software failure. The conventional software defect prediction models predicted the occurrence but not when the occurrence appeared. The wide range of experiments on existing techniques calculates the comparative results of the various software failure prediction and its limitations. Finally from the limitation identified from the existing works, further research work can be carried out with higher true positive rate while predicting the software faults with help of gene model. In addition, software reliability is enhanced when the fault in software system is predicted at earlier stage.

REFERENCES

Arunima Jaiswal, &. R. (2016). Software Reliability Prediction Using Machine Learning Techniques. International Journal of System Assurance Engineering and Management, Springer , 436, 1-15.

Barraza, N. R. (2015). A Parametric Empirical Bayes Model to predict Software Reliability Growth. Procedia Computer Science, Elsevier, 62, 360–369.

Changxing Zhu, H. Z. (2015). LSSVM-Based Rock Failure Criterion and Its Application in Numerical Simulation. Mathematical Problems in Engineering, Hindawi Publishing Corporation , 1-13.

Chao-Jung Hsu (2014). Optimal Weighted Combinational Models for Software Reliability Estimation and Analysis. IEEE Transactions on Reliability, 63 (3), 731- 749.

Dae-Young Kim (2016). Failure prediction of AZ31B magnesiumalloy sheet based on a micromechanical void model incorporating the asymmetric plasticity constitutive law. International Journal of Plasticity, Elsevier, 1-48.

Elahe Habibi (2015). Event-driven web application testing based on model-based mutation Testing. Information and Software Technology, Elsevier, 67, 159–179.

Feras A. Batarseh (2015). Predicting failures in agile software development through data analytics. Software Quality Journal, Springer , 1-18.

Fevzi Belli (2015). Exploiting Model Morphology for Event-Based Testing. IEEE Transactions on Software Engineering, 41 (2), 113 - 134.

Hellmich (2016). Statistical inference of a software reliability model by linear filtering. Journal of Statistics and Management Systems,Taylor and Francis Online, 19 (2), 163-181.

Houssany (2012). Microprocessor Soft Error Rate Prediction Based on Cache Memory Analysis. IEEE Transactions on Nuclear Science, 59 (4), 980 - 987.

Ilenia Fronza (2013). Failure prediction based on log files using Random Indexing and Support Vector Machines. Journal of Systems and Software, Elsevier , 86 (1), 2-11.

Julio Fernandez-Ceniceros (2014). Soft Computing Metamodels for the Failure Prediction of T-stub Bolted Connections. International Joint Conference SOCO’14, Springer , 299, 41-51.

Lorenzo Strigini (2013). Software Fault-Freeness and Reliability Predictions. Computer Safety, Reliability, and Security, Springer , 8153, 106-117.

M. Ben Hassine (2014). Time to failure prediction in rubber components subjected to thermal ageing: A combined approach based upon the intrinsic defect concept and the fracture mechanics. Mechanics of Materials, Elsevier , 79, 15-24.

Manjula Gandhi Selvaraj (2015). Predicting Defects Using Information Intelligence Process Models in the Software Technology Project. The Scientific World Journal, Hindawi Publishing Corporation, 2015, 1-6.

Marcello Cinque (2016). Characterizing Direct Monitoring Techniques in Software Systems. IEEE Transactions on Reliability, PP (99), 1-17.

Marcello Cinque (2012). Event Logs for the Analysis of Software Failures: a Rule-Based Approach. IEEE Transactions on Software Engineering, 39 (6), 806 - 821.

Mark D. Syer (2015). Replicating and Re-Evaluating the Theory of Relative Defect-Proneness. IEEE Transactions on Software Engineering, 41 (2), 179-197.

Michael Grottke (2015). Recovery From Software Failures Caused by Mandelbugs. IEEE Transactions on Reliability, 65 (1), 70-87.

Ping Jiang (2014). Weibull Failure Probability Estimation Based on Zero-Failure Data. Mathematical Problems in Engineering, Hindawi Publishing Corporation, 2015, 1-8.

Roberto Baldoni (2012). Online Black-Box Failure Prediction for Mission Critical Distributed Systems. Computer Safety, Reliability, and Security, Springer, 7612, 185-197.

Subhashis Chatterjee (2016). A new fuzzy rule based algorithm for estimating software faults in early phase of development. Soft Computing, Springer, 20 (10), 4023–4035.

Vahid Behbood (2015). Multi-Step Fuzzy Bridged Refinement Domain Adaptation Algorithm and Its Application to Bank Failure Prediction. IEEE Transactions on Fuzzy Systems, 23 (6), 1917 - 1935.