International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 6, Issue 3, May- June 2017 ISSN 2278-6856

A COMPARISION STUDY ON RELIABILITY BASED TESTING TO MINIMIZE VARIANCE

Anubha Arpan[1] , Mitrabinda Ray[2]

[1]Anubha Arpan, Institute of Technical Education and Research, S'O'A University
Bhubaneshwar 751030, Odisha, India

[2]Mitrabinda Ray, Institute of Technical Education and Research, S'O'A University
Bhubaneshwar 751030, Odisha, India

Abstract

Reliability is an anticipation of failure-free actions of a system over a stated time in a stated environment for a stated purpose. Blackbox testing is used for reliability estimation where the code is frozen. Reliability testing provides mechanism for making quantitative decision on reliability, about a process or processes . The main objective of this paper is to minimize variance in software reliability estimation.Reliability of a given software is inversely proportional to the variance.Various methods for estimating reliability are Operational Profile, Markov Chain, Controlled Markov Chain, Hidden Markov Chain etc.These methods are used in different types of testing strategy such as Operational Profile testing, Usage based testing, Adaptive testing, Modified adaptive testing etc.

Keywords: Adaptive Testing, Hidden markov chain, Markov Chain, Software Reliabilty, Statistical Testing.

1. Introduction

Reliability is used to check weather the software is performing as specified in specified environment and in specified time or not.The main objective of this paper is to compare different statistical methods of reliability estimation by using variance in reliability estimated by different methods.The reliability is one of the most important factor to be taken care of for designing any software.The reliability of the software can be estimated by introducing fault in design or implementation phase[1] and the other method for reliability improvement is fault detection and removal by freezing the code.Frozen code indicates code should not altered during testing[2]. The software testing is done for reliability assessment and improvement by using different statistical and probabilitic methods.The statistical testing is a blackbox testing technique in which the sequence of inputs are generated using probabilistic distribution. The probability distributions is anticipated with the help of usage of the software used for test[3].Statistical testing is used because of the fact that it gives quantitative decision for weather to accept or reject the hypothesis.There are different methods which can be used under statistical testing i.e.,Operational Profile, Markov Chain, Controlled Markov chain, Hidden Markov Chain.With the help of above mentioned methods test sequence are generated for reliability estimation is digramatically explained in figure 1.

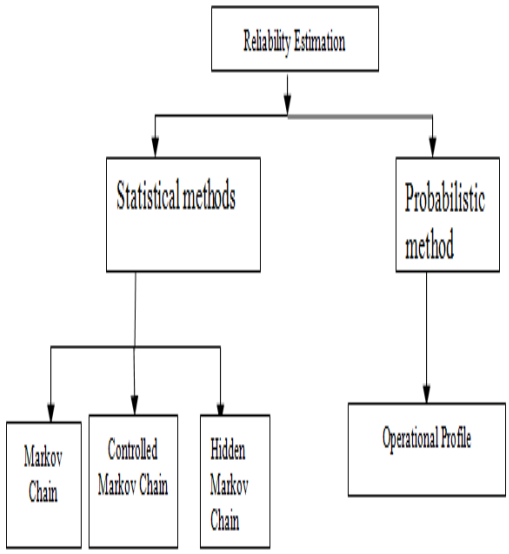

Figure 1:Reliability estimation methods overview

2.Reliability Estimation Methods:

Statistical testing uses mean, standard deviation, variance using different input conditions to estimate the reliability of the software. Statistical testing focuses on how fault in the program affects the reliability of the software. Different researchers have proposed various statistical methods for reliability estimation discussed below.

2.1 Operational Profile

The operational profile is measurable description of how the system will be used and all the profiles are together used to predict the probability of actual behavior of the system. With the help of operational profiles the test cases are prepared for all types of inputs[5]. It is observed that operational profile when used used with other testing strategy reduce the cost of maintainance by 10 times. In J. Musa's paper[5] flow of creating an Operational Profile is described which is explained in figure 2.The top four profile mentioned in the figure 2 are estimated based on the theoretical knowledge but last profile I.e., the operational profile is developed based on the actual implementation of the software.

Customer profile: Customers are those who purchases the software but it is not necessary that the software is used by them only. If a 10 company buys a particular software then the companies having same number of employee may use the software in similar way accordingly the customer profile is developed any other condition too.

User Profile: User group includes the actual user of the software, the occurrence probability of user profile is by dividing the users according to the estimated customer profile.

System-mode profile: The occurrence probability of the set of functions defined in the design phase and operations in the implementation phase are estimated in this phase.

Functional profile: It includes the occurrence probability of each function with respect to the given input or any explicit functions.

Operational Profile: Combining all above profile and by considering the actual implementation executions are divided into number of runs, input space is identified and partitioned and each input space is assigned with respective occurrence probabilities.

And finally the test cases are generated using the operational profile in such a way that there are minimum number of test cases used and all the expected failures are considered.

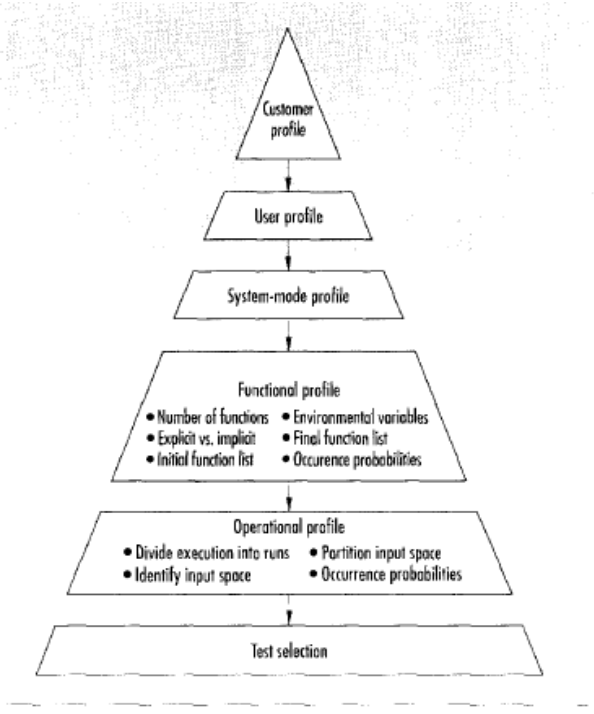

Figure 2 Development of operational profile step by step

Usage testing based on markov chain:
There are certain issues observe in the operational profile method first is the user may not use the software as assumed by the tester and secondly interaction of the software with the user is not initialized directly and it is not considered while explicitly developing the operational profile.

In operational profile inter-dependencies among the input given to the software and this problem is solved in Usage testing based on markov chain.

According to paper[3], A usage chain with the set of operations performed by the given software. The sequence of inputs must be maintained while creating the usage model. Each transition is assigned with system inputs and transitions probabilities. The transition probabilities are defined observing the real usage must follow the markovian property i.e., the current state does not depend on the previous state.

Example:
In the example of Usage Model given in figure 3 which is taken from paper[3], two conditions are used for giving input firstly, the current pointer location must be preserved to figure out the action of the "enter" key. Secondly, weather a project is defined to figure out which of the menu entry/entries are present.

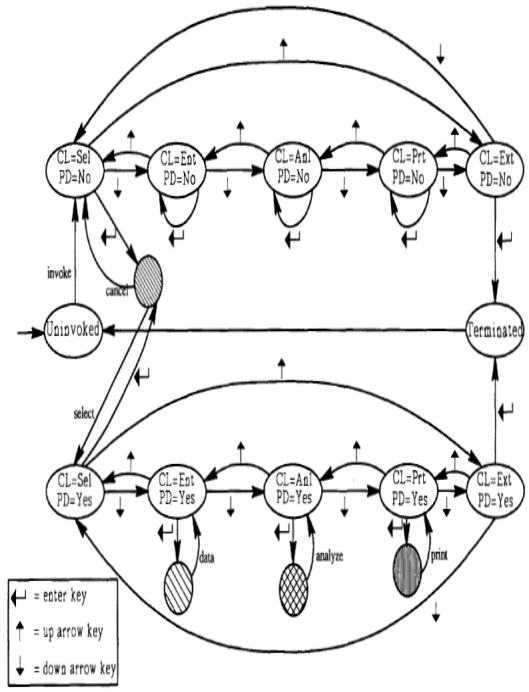

Figure 3 Usage chain for given example

2.2Controlled Markov Chain(CMC)
The software used for testing is treated as a controlled object and testing strategy works as a corresponding controller. The chain is terminated when number of input defects in the software is zero i.e., all the defects are detected and removed. Figure 4 describes the CMC approach for testing a software based on presence of number defects in the software. Adaptive testing uses CMC for reliability estimation[2].

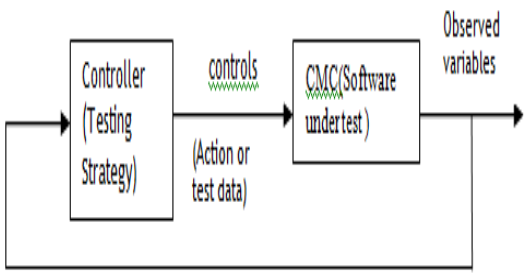

Figure 4 Software under test as CMC

2.3 Hidden Markov Chain:

Hidden Markov chain is used to model the bugs removal process. As the debugging process is considered as imperfect, i.e., new bugs may come out during the process of debugging[4]. The new bugs which imerged during bugs removal are modeled as hidden states. The hidden states are identified using Bayesian model.

In rest of the paper different reliability estimation methods are explained according to different authors with diagrammatical illustration in section 2, in section 3 the testing strategies which uses above mentioned reliability estimation methods is dicussed and section 4 covers the finding and argument of different authors in different papers [1-20].

3. COMPARISION FOR DIFFERENT RELIABILITY ESTIMATION METHODS

3.1. Usage based Markov Chain

In structural code based testing each line of program code should be executed atleast once,it tests all the reachable element with cost and time constraints.The limitations in using structural code testing is that it is too costly, it needs the tester to know the source code and above all the most disadvantageous thing is it does not consider the user specification while testing the software therefore the user may get some error after testing. This problem can be solved by using usage based testing. The usage model of a software distinguishes the practical use of the software considered for testing. This model provides knowledge about probability of occurances of each funtions to the tester and helps the tester to develop more efficient test cases.The markov usage chain constitutes a unique start state, set of intermediate states, a final state and number of transition arc wthich indicates the transition probabilities[21].Usage markov can be developed in two phases the first phase is structural phase which deals with the set of states and associated probabililites and the next is statistical phase define the unknown probabilities of the arc by ananlysing the other probabilities, these are assigned between 0 and 1[22]. According to paper[6] Markov Chain usage model for telephone use is discusssed as shown in 5.According to the discussed model [ON HOOK] is a initial state where the test cases starts and the chain terminates in [Exit] states.

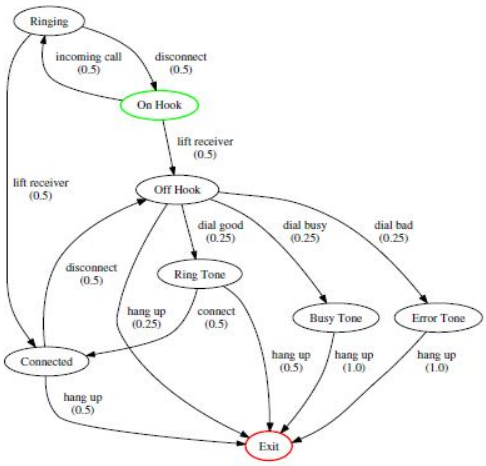

Figure 5 Usage model for telephone

The test cases are generated considering all the states of the model with minimal number of tests. Each transition are associated with transition probabilities to go to the next state. Suppose there are 100 telephone connected then there are 7^{100} states. Test cases are generated combinedly using all the states.

Example 3: Usage model for security alarm system is proposed in paper[19] depicted I figure 6 to estimate and assess reliability of the software. Mathematical calculations are also explained in paper[9] using probabilities of all transitions of states.

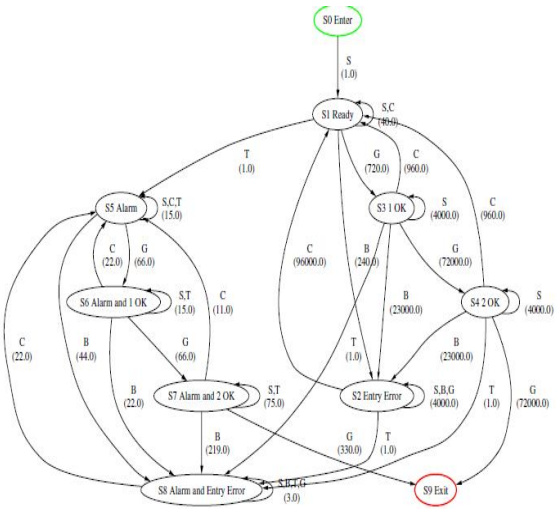

Figure 6 Security alarm model

3.2. Controlled Markov Chain(CMC)

The Markov Chain model may not perform as expected for the complicated software with more number of states two or more states takes same input class[].Then CMC is proposed which tests the software by considering the number of failure and remove them one by one. Adaptive testing uses CMC for test cases generation, the testing of software is treated as control problem. The software defects are detected and removed one by one forming a

markov chain. The defects are removed minimizing the cost of defect detection and removal. Adaptive testing follows a feedback path to develop new test case using the testing data history and the second feedback path for parameter estimation to calculate the reliability. Reliability is estimated with the help defect detection rate when all the defects are removed, finally the variance is calculated considering the different input conditions[2].

Modified Adaptive Testing uses failure division in reliability estimation and adjust the testing process according to importunity of the failure. The variance of the reliability estimator is diminished and accurateness of the software reliability is raised. In paper [7] improvement is done with respect to impact of failure classification on software reliability assessment. And the adaptive testing is improved according to importunity of the failures. Adaptive testing which uses CMC is explained with diagrammatic illustration given in paper[2] is shown in figure7.

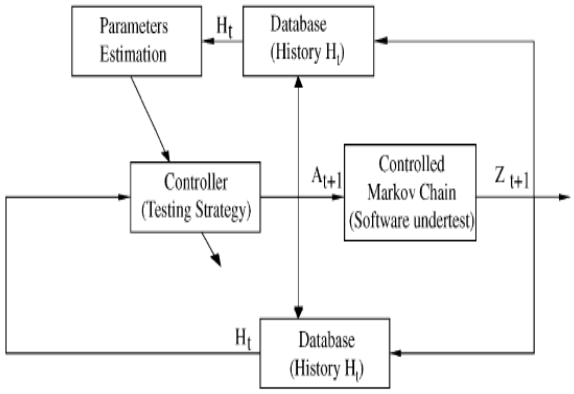

Figure 7 Adaptive testing using CMC

2.3. Hidden Markov Chain

The failure rate along with hidden defects are considered using Hidden Markov Chain process with transition probabilities with finite number of states. The hidden states are estimated using Bayesian method with the help of Markov Chain Monte Carlo(MCMC)[3] simulator. If software failure is observed for test stages x(n=x_1, x_2, x_3… x_n)then the posterior distribution for all unrecognized quantities is figured out using the likelihood function of the failures. The MCMC uses conditional probabilities for each states with respect to the likelihood function to explore the hidden states in the markov chain[3].

3.TESTING STRATEGIES FOR RELIABILITY ESTIMATION

The reliability of the system basically depends on two factors firstly on reliability of each component of the software and secondly on the probability distribution of input in each component of the system.The high is the reliability of the software it will give more accurate result accordingly. To improve the reliability of the software the test cases should be generated effectively to handle all type of faults like critical, non critical faults.The reliability estimation method should be made cost effective by minimizing the number of test cases and minimizing the cost of finding and removing each defects. The above mentioned reliability estimation methods are used by different testing strategies for test case generation and those test cases are used to handle the failures present in the software and finally estiamting the software reliability.There are various testing strategies using which reliability can be estimated proposed by different authors in papers[1-20]. The effectiveness is also proved by experimental illustrations.The different testing strategies are explained below.

3.1. Adaptive Testing(AT)

In AT, the software testing process is considered in the role of control issue and the correspondent testing approach is treated as a control policy or controller, and the Software to be tested act as a controlled item with contingency. Additionally, if an optimization (testing) aim is specified separately, then the test case selection problem is interpreted into an optimal control issue. In an optimal control problem, a dynamic system (controlled object) is defined which follow the Markov Chain with respect to some variable. The control policies can be decided using the history of system. The input space is divided into m parts and testing is done for each class to detect and remove the defects following the concept of CMC. The unknown parameter involved in process of testing are estimated online.[9]

3.2. Adaptive testing-gradient-descent(AT-GD)

To minimize the computational overhead of decision-making AT-GD is used. the executed AT strategy in practice varies from its theoretically explained design that guarantees AT's local optimality. The goal of this paper[9] is to explore the asymptotic behavior of AT to improve its global performance without losing the local optimality. Therefore, a new AT strategy named Adaptive Testing with Gradient Descent method (AT-GD) is proposed. Theoretical analysis indicates that AT-GD, a locally optimal testing strategy, converges to the globally optimal solution as the assessment process proceeds. Experiments and simulation are performed to check AT-GD's effectiveness and efficiency and sensitivity analysis is also done. In order to preserve the local optimality while achiving global optimization AT-GD is designed[9].

3.3. Random Testing

In Random testing it is assumed that the software to be tested is designed by a binomial probability distribution which varies with respect to the equivalence class of selected test cases. If a test case of equivalence class i is selected, then a software failure is observed with probability of defect detection rate in class i and no software failure is observed in class i in p(1-defect detection rate). This probability distribution is used to determine either the defect is detected or not each time. The operational profile is used each time as the test profile and variance is calculated after finding the reliability.[2]

3.4. Reliability Improvement And Assessment Testing(RELAI)

RELAI is an Operational testing.RELAI uses improved approach for reliability estimation and reliability assessment.It address the problem of low-occurance failure and inaccurate operational profile. RELAI adaptively test the software and as soon as the high –failure occurance area of input is done with test case selection it proactively moves to low-failure occurance zone. In the previously used test strategies[2,9] the codes are remain unaltered during testing but this assumption is removed in RELAI[10].

4.FINDINGS AND ARGUMENT

The finite state, discrete parameter, time homogeneous Markov chain represents a practical option for random testing in the development and analysis of usage models and automatic test input generation. There are so many successful applications of Markov chain usage models for discussed by different researchers in [1, 6], involving both real-time embedded systems and user based applications. Markov chain is not suitable for complex software where there are multiple states with same input conditions. In paper[6], Markov model for telephone system is described.

In paper [8] experiment is done with a software program using different testing strategies to find reliability. In an experiment the result of Operational Profile testing. Adaptive testing and random testing results are compared in different phases of testing. The mean, offset, and variance are compared with respect to the reliability estimator are compared .Adaptive testing outperforms the other two testing strategies in all the three phases.

According to analysis done in paper [2] comparison is done between adaptive testing and random testing and optimal testing and it is calculated that adaptive testing has minimum variance.

AT-GD is proposed after adaptive testing [9] and experimentally proved that AT outperforms Operational testing and random testing but AT-GD gives more accurate result as compared to AT.

According to paper[10] RELAI is proposed in which experiment is done not only by considering the faults which are visible but takes into account the region where fault may occur. It takes care of the error in profile and also minimizes the number of tests required to minimize the cost.

5. CONCLUSIONS

The testing based on the model of usage of the software is used for testing with the help of their transition probabilities in the state transition diagram. In usage model testing model once designed need not be changed as the operations are done on the test cases, this is proved to be an efficient method for testing but for complex software it is difficult to design models as the input conditions may remains same for more than one input class. Then CMC is proposed for testing which assumes code to be frozen and testing is done based on the defects present in the software. CMC moves from one state to next state by removing the defect detected following the feedback path for test case selection and parameter(defect detection rate, failure rate) for testing is estimated online. Among all the testing strategies discussed such as AT which tests the software minimizing the number of test cases using the test data history, AT-GD which aims to preserve the local optimality while achieving the global optimaliy, MAT tests the software with respect to critical and non critical failures, RELAI uses CMC method for testing and finding reliability, adaptive testing using CMC method is proved to be most efficient method in calculating and assessing reliability of a software. The comparative study is done in different papers among the testing strategies.

References

[1] M. Kooli, H. Kaddachi.,D. Natale, G. Bosio, P. Benoit, L. Torres," Computing reliability: On the differences between software testing and software fault injection techniques", Microprocessors and Microsystems,vol. 50, pp. 102-112, 2017.

[2] K. Y. Cai, Y. C. Li, K. Liu, "Optimal and adaptive testing for software reliability assessment" Information and Software Technology, vol. 46(15), pp. 989-1000, 2004.

[3] A. Pievatolo, F. Ruggeri., R. Soyer " A Bayesian hidden Markov model for imperfect debugging", Reliability Engineering & System Safety, vol. 103(2), pp. 11-21, 2012.

[4] J. A. Whittaker,M. G. Thomason , " A Markov chain model for statistical software testing", IEEE Transactions on Software engineering ,vol. 20(10), pp. 812-824, 1994.

[5] J. Musa , "Operational Profiles in Software Reliability Engineering," IEEE Software, vol. 10(2), pp. 14-32, 1993.

[6] S. Prowell, "Using Markov chain usage models to test complex systems" in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05), Big Island, Hawaii, USA, 2005, p.318c.

[7] J. Lv , H. Hu, K. Y. Cai, T. Y. Chen, "Adaptive and random partition software testing" IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 44(12), pp. 1649-1664, 2014.

[8] K. Y. Cai, C. H. Jiang , Hu H., C. G. Bai , " An experimental study of adaptive testing for software reliability assessment", Journal of Systems and Software, vol. 81(8), pp. 1406-1429, 2008.

[9] J. Lv, B. Yin, K Y. Cai, "On the asymptotic behaviour of adaptive testing strategy for software reliability assessment", IEEE transactions on software engineering, vol. 40(4), pp. 396-412, 2014.

[10] D. Cotroneo, R. Pietrantuono, S. Russo ," Relai testing: A technique to assess and improve software reliability", IEEE Transactions on Software Engineering, vol. 42(5), pp. 452-475,2016.

[11] T. Y. Chen, F. C. Cuo, H. Liu, W. E. Wong, "Code coverage of adaptive random testing", IEEE Transactions on Reliability, vol. 62(1), pp. 226-237, 2013.
[12] J. Lv , Yin B B., K. Y. Cai, "Estimating confidence interval of software reliability with adaptive testing strategy",Journal of Systems and Software, vol. 97(2), pp. 192-206, 2014.
[13] A K. Jena, S. K. Swain,D. P. Mohapatra, A novel approach for test case generation from UML activity diagram "In Issues and Challenges in Intelligent Computing Techniques (ICICT)", 2014 International Conference on IEEE, pp. 621-629, 2014.
[14] M. J. Escalona, J. J.Gutierrez, M. Mejas, G. Aragn, I. Ramos J. Torres, F. J. Domnguez, "An Overview on Test Generation from Functional Requirements", Journal of Systems and Software, vol. 84(8), pp. 1379-1393, 2011.
[15] P. Cao, Z. Dong., K. Liu, K. Y. Cai, "Quantitative effects of software testing on reliability improvement in the presence of imperfect debugging", Information Sciences, vol. 218, pp. 119-132, 2013.
[16] D. Homm, J. Eckert, R. German" CenUMs–concurrency enhanced usage models for statistical testing of complex systems with concurrent streams of use" Science of Computer Programming, vol. 132, pp. 173-189, 2016.
[17] M. Cinque., D. Cotroneo,R. Della Corte, A. Pecchia," Characterizing Direct Monitoring Techniques in Software Systems" IEEE Transactions on Reliability ,vol. 65(4), pp.1665-1681, 2016.
[18] T. Y. Chen, F. C. Cuo, H. Liu, W. E. Wong, "Code coverage of adaptive random testing. IEEE Transactions on Reliability", vol. 62(1), pp. 226-237, 2013.
[19] S. J. Prowell , J. H. Poore, "Computing system reliability using Markov chain usage models", Journal of Systems and Software, vol. 73(2), 219-225, 2004.
[20] Q. Li., C. Mao "Considering testing-coverage and fault removal efficiency subject to the random field environments with imperfect debugging in software reliability assessment", In Software Reliability Engineering Workshops (ISSREW), 2016 IEEE International Symposium on IEEE, pp. 257-263, 2016.
[21] G. H. Walton, J. H. Poore , Trammell C. J., "Statistical testing of software based on a usage model" Software: Practice and Experience, vol. 25(1), pp. 97-108, 1995.
[22] J. H. Poore,G. H. Walton, J. A. Whittaker, "A constraint-based approach to the representation of software usage models", Information and Software Technology, vol. 42(12), pp. 825-833,2000

AUTHORS

Anubha Arpan M.Tech scholar of computer science and engineering branch at Institute of Technical Education and Research.

Mitrabinda Ray Asst. Professor of computer science and engineering branch at Institute of Technical Education and Research.