Journal of Advances in Science and Technology
Vol. 18, Issue No. 2, September–2021, ISSN 2230-9659

A Study of Certain Software Reliability Related Problems

Sharad Kumar Dubey[1]*, Dr. Rajeev Yadav[2]

[1] Research Scholar, Shri Krishna University, Chhatarpur M.P.

[2] Professor, Shri Krishna University, Chhatarpur M.P.

Abstract - Software dependability is a crucial component of software quality. Software is rigorously examined and faults are fixed before it is released into the market. Every software business aspires to create error-free software. A system's availability refers to how well it is operational and accessible when it is needed for usage, while reliability refers to a program's ability to carry out its necessary duties. This implies to the user that a system is more trustworthy if it successfully completes the duties put out to it. Additionally, the research included topics such as Model Validation, Comparison Criteria, and Data Analyses, as well as Model Comparison Results, Model Parameter Estimation Results, Software Testing, Software Bugs, and Software Reliability.

Keyword - software reliability, SRGM

----------X----------

INTRODUCTION

A known and acknowledged reality is that software solutions are critical in all walks of life. Software systems research and industry have consistently and tirelessly given mankind some remarkable software products that have squeezed the whole globe into the grasp of the common man and have brought the entire humanity closer together to exchange experiences on a globalized platform. The human race has already made great strides in research by conquering the moon, Mars, and beyond with the help of really amazing and very complex solutions. As a result, mankind has been able to successfully launch multipurpose satellites, space shuttles, and other spacecraft in order to better understand and forecast the happenings in the universe with the ultimate goal of maintaining life on Earth and exploring new places on other planets for the possible existence of life. Even while this technology is required for every operation, it is not enough to just have a high level of accuracy. Traveling millions of light-years is now something we can envisage. In contrast, real-time and mission-critical systems need hefty development expenses. Leaving even the slightest amount of leeway for mistake in safety-critical apps puts users' lives at grave danger.[1]

Reliability along with reliability, flexibility, efficiency, serviceability, capability, installation capacity, maintenance capacity and documentation are an essential feature of software quality. The software reliability is described as "the possibility of failure-free operation of software over a specified period of time in a specified setting" according to ANSI (American National Standards Institute). Computer Software Trustworthiness is difficult to obtain since software is also highly technical. While it's impossible to achieve a certain degree of reliability with all extremely complicated structures, like applications, device engineers prefer to move complications through their software layer, with the quick growth of systems size and simple to do so with a software update. Although software complexity is inversely associated with software stability, it is directly linked to other major software quality variables, especially functionality, capacity etc. [2]

Software Reliability

In addition to usability, performance, serviceability, capability, install ability and documentation's importance to software quality is reliability. Software Reliability is defined by the American National Standards Institute (ANSI) as "the likelihood of failure-free software execution over a specific amount of time in a specified environment". Software The difficulty of achieving reliability is due to the enormous complexity of software. Because of the fast increase in system size and the ease with which software can be upgraded, system engineers prefer to push complexity into the software layer, despite the fact that this makes it difficult to achieve a particular level of dependability. Complexity of software affects software dependability negatively, but it also has a positive impact on other crucial aspects of software quality, such as its capacity to perform many functions. By emphasizing these functions, software will become more sophisticated.[3]

Software Bugs

High-availability server applications have necessitated an increase in reliability. In the event of system failures, productivity and profitability might be severely harmed, resulting in a considerable decrease in system availability. For a financial business, one hour of downtime costs more than $6 million, according to a survey from Gartner Group. Many major tragedies have been caused by software defects in crucial systems, including aircraft accidents, nuclear reactor shutdowns, and more. Unfortunately, flaws in software are still a common occurrence. Software bugs have been shown to be one of the most common causes of system failures in a number of different studies. Gray's research from 1986 suggests that 25% of Tandem system failures were caused by software defects. More than 40% of all system failures are the result of software mistakes, according to a study from the year 2000. Each year, software flaws cost the U.S. economy around $59.5 billion, or 0.6% of GDP, says the National Institute of Standards and Technology (NIST). Most software businesses devote considerable resources to software testing and debugging in order to improve the overall quality of their products. Research suggests that 50-80% of development and maintenance time is devoted to reducing the amount of faults in the code that is actually delivered.[4]

Software Testing

In order to determine if a programme or system has the desired features or capabilities, it must undergo software testing. Software testing is a vital part in ensuring the quality of software. Instead of showing that a software works perfectly, good testing should reveal that it has flaws. Every stage of the Software Development Life Cycle must be tested to ensure the quality of the product (SDLC). Discipline is being imposed as a result of this strategy. The software is tested as it is built and integrated to the developing system to verify that it works properly and effectively

LITERATURE REVIEW

Y. Geetha Reddy, Dr. Y Prasanth (2020) it is believed that software systems developed utilising statistical learning models would be reliable. However, a large percentage of flaws go unnoticed in small and medium-sized implementations. During the product development and testing phase, a wide variety of trust tests are employed to detect software errors. Real-time trust assessment is based on new defects discovered throughout the app examination and repair process. It is critical to manage very trustworthy findings in contemporary SRGMs and these models are not important to mathematical dependencies and assessments of independence. The suggested methodology uses a novel quartile density distribution model that focuses on the dependability prediction rate. In terms of skewedness and weakness, this model performs better than classic growth models, according to experimental investigations. [5]

Deepak Kumar, ShubhraGautam (2018) In today's society, with the growing demand for information solutions, the most efficient systems must be built. Software stability is the likelihood that a device is free of malfunction under any circumstances over a specified time. To calculate the consistency of the app, software Reliability Growth Models (SRGMs), Many SRGMs accept the one-stage approach of program stability. However, some researchers regard it as a twostage method for observing and removing errors. Furthermore, the literature examines the incomplete removal of program failure. Two methods of debugging may take place, i.e. imperfect deletion and error generation. In this article we provide two new SRGMs that use a learning feature to remove defective defects and generate errors. Actual software data sets and frameworks are validated in conjunction with present models. Based on the importance of the parameters, the proposed model may even be reduced to current model. [6]

Da Hye Lee, In Hong Chang, Hoang Pham and Kwang Yoon Song (2018) "A Software Reliability Model Considering the Syntax Error in Uncertainty Environment, Optimal Release Time, and Sensitivity Analysis**"** The software developers have set themselves the aim of developing high-quality and dependable products. Software has grown complicated in the last decades and thus reliable software solutions are harder to create. Failures in software can entail severe social or economic damage therefore software dependability is essential. To calculate software reliability, software reliability growth models (SRGMs) have been employed. In this study we provide a novel dependability model of software and compare it with a number of non-homogeneous Poisson process models (NHPPs). We also evaluate the fitness of existing GMRs on the basis of eight criteria using real sets of data. The findings enable us to identify the optimum model. [7]

Song, Kwang& Chang, In & Pham, Hoang (2017) "A Software Reliability Model with a Weibull Fault Detection Rate Function Subject to Operating Environments" These systems are utilized in field settings that are same or similar to those used in development test environments in the introduction of software systems; however, they may also be employed at many places that vary from the environment in which they have been created and tested. For many reasons, such as a particular environment or a code fault location, it is difficult to increase software dependability. We present a novel model of software dependability, in which the unpredictability of operating environments is taken into consideration. For the suggested paradigm, the explicit mid-value solution is given. Examples of the fitness of various current NHPp models are given to demonstrate the fit of all models based on two sets of failure data gathered from software applications. The models are based on two different sets of failure data. The findings indicate that the suggested

model more closely matches the data than other current NHPP models.[8]

JiajunXu and Shuzhen Yao (2016) The majority of the non-homogeneous Poisson process (NHPP) software reliability growth models (SRGMs) commonly presume flawless or incomplete debugging. In the production and test process, however, environmental conditions generate considerable complexity for SRGMs. In quantifying uncertainties associated with a method of ideal or imperfect debugging, we propose a new NHPP model, focused on a partial differential equation (PDE). As a noise of random correlation, we reflect environmental insecurities collectively. In the modern stochastic system, the complete statistic knowledge of, for instance, the debugging method may be determined (PDF). The suggested model is similar to observation by some similarities with historical evidence and current methods, such as the classical NHPP model. [9]

Fan Li and Ze-Long Yi (2016) The SRGMs based on a non-homogeneous Poisson process (NHPP) are commonly utilized in describing the behavior of stochastic failure and evaluating software system reliability. Checking and fault interdependency perform important roles with these models. Taking into account the power law feature of test effort and the inter-dependence of multiple-generation faults, we suggest an updated SRGM to rethink the trust of open source systems (OSS) and then to use various real-world data to verify the output of that model. Our observational tests demonstrate that the model blends well into the failure data and has a very strong potential for prediction. In terms of test expense and reliability criteria, we also officially study the optimal product release strategy. Through carrying out sensitivities research, we will notice that the right period to release the program would be seriously postponed and further money wasted in testing the software if the testing impact or the failure interdependence is ignored. [10]

Mohammed Alweshah, Walid Ahmed, HamzaAldabbas (2015) Reliability growth model prediction and identification and removal of mistakes for software and software engineering engineers and project managers is both a need and a difficulty. The ability to estimate the amount of software bugs greatly aids in deciding the delivery date of the software and in successful project capital management. The cumulative failures in the test phase can be evaluated by two or three parameters in most growth models. In recent years there has been an increased interest in using evolutionary calculation to solve prediction and modeling problems. This paper discusses the usage of genetic programming (GP) as an instrument for designing growth models that can reliably estimate the amount of software errors at an early stage of the research process. [11]

Chiu, Kuei-Chen (2013) In recent decades, SRGM models have been developed for the estimation of device stability during testing/debugging. The majority of the models is built on the NHPP and a test behavior of S or exponential form is normally presumed. Chiu et al. (2008) presented an SRGM which takes into account the study results and can reasonably explain both S-shaped and exponential behaviors. In this document both linear and empirical study results are considered in an SRGM in order to enhance the Chiu and others model (2008). The learning effects are dependent on the duration of the research and analyze what and what effects the program implementation will have. This analysis further explores the feasibility of the proposed R Square models (Rsq) and contrasts the findings with the other models through the use of four actual datasets. The suggested models concurrently consider the results of continuous, longitudinal and exponential learning. [12]

P. ARUN BABU (2013) In terms of functionalities, expense, flexibility, maintenance and reusability software-based systems have many advantages over hardware-based systems. Code is, however, apt to crash. Poorly written security-critical software may trigger disastrous breakdowns and life threats. Technology that is crucial for protection must then be properly checked and it is essential to study the likelihood of software errors. Software usability queening is perceived to be an outstanding issue; current methods and templates are not suitable with regard to safety implementations and have expectations and limitations. It also requires researching the variables that may influence software stability in order to develop stable software. [13]

Mohd.Anjum, Md. AsrafulHaque, Nesar Ahmad (2013) Many SRGMs have been analyzed to calculate growth in program reliability. Many SRGMs were analyzed. For researchers in the field of program usability, the choice of optimal SRGMs for use in a given case has been an area of concern. For each comparative parameter, all current methodologies apply the same weight. In fact, however, all parameters have not equal priority in the calculation of reliability. In this article, the issue of efficiency analysis of different non-homogeneous Poison process (NHPP) models is posed with a statistical approach focused on weighted parameters. It is very easy and needs to be measured less. A series of 12 comparative criteria were established and the software reliability development models proposed over the past 30 years were allocated to different weights for each criterion. Results from case studies suggest that the weighted system of importance criterion provides a very promising strategy for comparing app reliability development model. [14]

Peter H. Feiler John B. Goodenough Arie Gurfinkel Charles B. Weinstock Lutz Wrage (2012) the scale and sophistication of software-related devices, such as rotorcraft and others, has grown exponentially. Develop then testing has left them unaffordable to build and qualify for the current

software engineering procedure. This study explores the difficulties in the certification and results of many studies undertaken by government and business. It identifies a number of root causes and provides a reliability validation and improvement framework integrating several of the recommended technological solutions: a verification of the formalized needs; an architecture-centered, model-based engineering approach which detects early system problems through analysis; It also forms the foundation for a series of metrics to boost cost-effectiveness to meet emerging software sophistication, trustworthiness, and cost metrics challenges.[15]

Chao-Jung Hsu, Chin-Yu Huang, Jun-Ru Chang (2011) many software reliability models have been proposed for estimating the growth of software products in the last three decades with different parameters that represent different testing properties. We have found that the fault reduction factor (FRF) proposed by Musa is among the most significant parameters for software reliability development. FRF is usually defined as the net failure reduction ratio. Due to multiple external causes, such as imperfect debugging and debugging time lag, FRF may be affected throughout software testing. Thus, we first examine some real facts in this paper in order to track the FRF patterns and recognize FRF as a time-changing feature. We research in addition how time variable FRFs are integrated into growth modeling for program reliability. Some experimental findings suggest that the models suggested will increase the exactness of the program reliability assessment. Sensitivity analysis focused on cost and durability criteria was addressed for different optimum release dates. The analysis shows that the modification of FRF value can affect period of release and costs of production. [16]

Chiu, Kuei-Chen & Huang, Yeu-Shiang&Lee, Tzai-Zang, (2008) Reliability improvement in program reliability in the testing/debugging process has been studied over the past three decades. Most models were built based on the NHPP (Non-homogenous Poisson Process) and are typically supposed to be S-shaped or exponential-shaped. These models may only, sadly, be ideal for basic software failure data, thus restricting application coverage. Therefore, we considered efficiency of testing and debugging, from the point of view of learning impacts, which can impact the process of software reliability development, that not only concerned the skill of test personnel, but also the learning effect derived from testing and debugging codes. Simultaneously, and the experiment findings are fit, the suggested solution will reasonably characterize the S-formed and exponential forms of comportment. A comparative study was also undertaken to assess the feasibility of the proposed model and other models of program failure. Finally, we propose an acceptable release strategy for software. [17]

P.K. Kapur, V. B. Singh, Sameer Anand, V. S. S. Yadavalli (2008) Many software reliability growth model (SRGM) models focused on a non-homogenous Poisson (NHPP) mechanism have been established with the expectation that an FDR and a failure prediction process depending on the residual fault material are usable. This paper uses a particular method for model creation to create an SRGM focused on NHPP. In this case, not only the content of the residual fault but also the test period is the fault detection mechanism. In software growth, the fault identification rate doesn't continue throughout the whole test phase but varies due to fluctuations in resource distribution, fault density, operating setting and test strategies (called the change-point). The FDR is described here as a test time feature. The suggested model often includes the test effort with the change-point principle to solve the issues of rushed software efforts and provides project management with a strategy to coordinate the test effort and stability to achieve the optimal degree of reliability. [18]

METHODOLOGY

The role of learning remains important in understanding the variations in testing process. Learning manifests as an S-shaped behavior of mean value function (MVF) or the testing effort (TE) function. Learning plays important role on fault detection rate also. In this study, the variations in fault detection rate based on gains in learning are considered as S-shaped learning based rate function. The main objective of this research is to carry out study and development of some improved or new software reliability growth models under the analytical framework of NHPP. The development of SRGMs is done incorporating the concepts of learning and change-point analysis. The models are checked against the statistical structures of different datasets and statistical tests are used to validate the proposed growth models.

Two types of learning have been identified and two new models have been proposed considering these two types of learning. First one of the two proposed models incorporates these two types of learning and second proposed model extends the first proposed model by incorporating a negligence factor, in addition to two type learning. The negligence factor represents failure on part of testers to apply the learnt patterns due to negligence. The proposed models have been statistically analyzed and validated using actual software failure datasets. This study also considers fault detection rate functions in the general NHPP imperfect debugging methodology. Two new imperfect debugging models under learning-factor based fault detection rate (LB-FDR) have been instituted differing under the assumptions of the form (exponential or linear) of fault content (FC) functions which involve a fault introduction factor representing imperfect debugging. The first proposed fault detection rate model is based on exponential fault content (EFC) function and the second proposed fault detection rate model is based on linear fault content (LFC) function. The proposed models have been statistically analyzed and validated using actual software failure datasets. Further, this study considers the problem of change-point which under

practical settings could represent an abrupt gain in learning effect due to introduction of new testing techniques, tools, strategies etc. A new distribution-based model incorporating change-point under imperfect debugging settings is proposed, statistically analyzed and validated.

DATA ANALYSIS

1. Model Validation, Comparison Criteria and Data Analyses

We used a genuine software data set to demonstrate the estimating technique and use of the SRGM (existing and planned). The statistical tool SPSS was used to estimate the models' parameters, and change-point analyzers were used to assess the data sets' change-point.. Data analysis of genuine software data sets was used to demonstrate the estimate technique of the SRGM (existing and planned, respectively) and its application. The statistical tool SPSS was used to estimate the model parameters and the change-point analyzer was used to assess the datasets.

1.1 Data set1(DS-1)

Faults were found in the initial data set (DS-1) after 35 months of testing a radar system of 124 KLOC in size. Brooks and Motley provide the information used here (1980). The 17th month marks the beginning of a new trend in this dataset.These findings may be read in Table , which shows parameter estimate and comparison criteria for DS-1 of all the models in the study. The table shows that SRGM-3 and SRGM-5 have higher R2 values and lower MSE, Bias, Variation, and RMSPE values than other models, indicating that they give superior goodness-of-fit for DS-1 than the other models.

1.2 Data set 2(DS-2)

When real-time command and control was tested for 19 weeks, 328 defects were uncovered in the second data set (DS-2). Ohba provided the referenced information for this (1984). This data set's change point is the sixth week. Table 3.4 and Table 3.5 provide the parameter estimate and comparison criteria findings for DS-2 of all the models included in the study. The table shows that the R2 values for SRGM-3 and SRGM-5 are higher and the values of MSE, Bias, Variation, and RMSPE are lower in comparison to other models, and the DS-2 model gives a superior match.

Table 1: Model Parameter Estimation Results (DS-1)

Models	a	b_1	b_2	μ_1	μ_2	σ_1	σ_2	α_1	α_2	λ_1	λ_2	k	β
SRGM-1	2705	.019	.023	-	-	-	-	-	-	-	-	-	-
SRGM-2	1639	.092	.095	-	-	-	-	-	-	-	-	-	-
SRGM-3	1321	.213	.211	-	-	-	-	-	-	-	-	-	25
SRGM-4	1303	.0007	.0007	-	-	-	-	-	-	-	-	2	-
SRGM-5	1298	-	-	11	15	5	2	-	-	-	-	-	-
SRGM-6	1350	-	-	-	-	-	-	.683	4.816	.031	.281	-	-

Table 2: Model Comparison Results (DS-1)

Models	R^2	MSE	Bias	Variation	RMSPE
SRGM-1	.97367	5608.09	-1.28E-05	75.9804	75.9804
SRGM-2	.98833	2486.007	-3.89E-06	50.5877	50.5877
SRGM-3	.99930	149.5196	-4.96E-07	12.4063	12.4063
SRGM-4	.99872	271.789	-22.08	25.0093	33.3660
SRGM-5	.99922	166.526	-0.0001	13.0928	13.0928
SRGM-6	.99700	731.7273	-2.85E-07	23.9072	23.9072

Table 3: Model Parameter Estimation Results (DS-2)

Models	a	b_1	b_2	μ_1	μ_2	σ_1	σ_2	α_1	α_2	λ_1	λ_2	k	β
SRGM-1	614	.040	.044	-	-	-	-	-	-	-	-	-	-
SRGM-2	401	.189	.169	-	-	-	-	-	-	-	-	-	-
SRGM-3	365	.232	.217	-	-	-	-	-	-	-	-	-	4
SRGM-4	372	.022	.018	-	-	-	-	-	-	-	-	1.625	-
SRGM-5	351	-	-	6	7	1	2	-	-	-	-	-	-
SRGM-6	411	-	-	-	-	-	-	1.349	1.872	.103	.15	-	-

Table 4: Model Comparison Results (DS-2)

Models	R^2	MSE	Bias	Variation	RMSPE
SRGM-1	.98812	122.6662	-6.55E-07	11.3789	11.3789
SRGM-2	99054	97.61703	5.32E-06	10.1508	10.1508
SRGM-3	99270	75.29769	3.40E-06	8.9152	8.9152
SRGM-4	99143	88.41156	-2.85E-05	9.6603	9.6603
SRGM-5	.99300	72.28211	0.0017	8.7348	8.7348
SRGM-6	.99100	97.23347	-0.0646	10.1311	10.1313

CONCLUSION

There are two basic and realistic models of increase in software dependability that combine two learning principles. In order to improve learning by technology investment, the learned lessons are taken into consideration as well as the autonomous learning. In addition to 2-type learning in the second model, a negligent element is a failure of the test testers to use the learned patterns in the error detection and repair process. The cost analysis for the suggested models may be expanded for this research and thus offer a rule governing when to cease the testing and release the program. In addition, its study of the inflection point may be studied. The environmental impact may be further explored on the modelling of learning effects. In addition, the suggested model may be analyzed

using a Bayesian or Quasi-Bayesian method with inadequate data and an unknown environment.

REFERENCE

1. Iannino A., Musa J.D., Okumoto K., Littlewood B. "Criteria for Software Reliability Model Comparisions", IEEE Transactions on Software Engineering, SE-10(6): pp687-691, 1984.

2. ArchanaKumar,Ph.D. Thesis "A Study in Software Reliability Growth Modelling Under Distributed Development Environment", Department of Operational Research, University of Delhi, Delhi, 2007.

3. Popstajanova, K. and K. Trivedi: "Architecture Based approach to Reliability Assessment of Software Systems", Performance Evaluation, Vol. 45, No.2, pp.179-204, 2001.

4. Yamada S, Ohba M and Osaki S, "S-shaped reliability growth modeling for software error detection", IEEE Transaction on Reliability, Vol. 32, pp.475–478, 1983.

5. Y. Geetha Reddy, Dr. Y Prasanth, "STATISTICAL QUARTILE DEVIATION-BASED SOFTWARE RELIABILITY GROWTH ESTIMATION MEASURE FOR RELIABILITY PREDICTION", Journal of Critical Reviews ISSN- 2394-5125 Vol 7, Issue 2, 2020

6. Deepak Kumar, ShubhraGautam" Flexible Software Reliability Growth Models Under Imperfect Debugging and Error Generation Using Learning Function", Journal of Management Information and Decision Sciences, 2018 Vol: 21 Issue: 1

7. Da Hye Lee, In Hong Chang, Hoang Pham and Kwang Yoon Song, Appl. Sci. 2018, 8, 1483; doi:10.3390/app8091483

8. Song, Kwang& Chang, In & Pham, Hoang. (2017). A Software Reliability Model with a Weibull Fault Detection Rate Function Subject to Operating Environments. Applied Sciences. 7. 983. 10.3390/app7100983.

9. JiajunXu and Shuzhen Yao, "Software Reliability Growth Model with Partial Differential Equation for Various Debugging Processes", Volume 2016, Article ID 2476584, 13 pages, Mathematical Problems in Engineering.

10. Fan Li and Ze-Long Yi , "A New Software Reliability Growth Model: Multigeneration Faults and a Power-Law Testing-Effort Function", Volume 2016, Article ID 9276093, Mathematical Problems in Engineering

11. Mohammed Alweshah, Walid Ahmed, Hamza Aldabbas,(2015) "Evolution of Software Reliability Growth Models: A Comparison of Auto-Regression and Genetic Programming Models", International Journal of Computer Applications

12. Chiu, Kuei-Chen, "A Discussion of Software Reliability Growth Models with Time-Varying Learning Effects", American Journal of Software Engineering and Applications, volume 2, Issue 3, June 2013, Pages: 92-104

13. P. ARUN BABU,(2013) "SOFTWARE RELIABILITY IN SAFETY CRITICAL SUPERVISION AND CONTROL OF NUCLEAR REACTORS" ACM Sigsoft software engineering notes, Volume 37, Issue 5.

14. Mohd. Anjum, Md. AsrafulHaque, Nesar Ahmad, "Analysis and Ranking of Software ReliabilityModels Based on Weighted Criteria Value", I.J. Information Technology and Computer Science, 2013, 02, 1-14

15. Peter H. Feiler John B. GoodenoughArieGurfinkel Charles B. Weinstock Lutz Wrage, "Reliability Validation and Improvement Framework", Research, Technology, and System Solutions Program, November 2012.

16. Chao-Jung Hsu, Chin-Yu Huang, Jun-Ru Chang, Enhancing software reliability modeling and prediction through the introduction of time-variable fault reduction factor, Applied Mathematical Modelling, Volume 35, Issue 1, 2011, Pages 506-521,

17. Chiu, Kuei-Chen & Huang, Yeu-Shiang& Lee, Tzai-Zang, 2008. "A study of software reliability growth from the perspective of learning effects," Reliability Engineering and System Safety, Elsevier, vol. 93(10), pages 1410-1421.

18. P.K. Kapur, V. B. Singh, Sameer Anand, V. S. S. Yadavalli, "Software reliability growth model with change-point and effort control using a power function of the testing time ", International Journal of Production Research 46(3):771-787

Corresponding Author

Sharad Kumar Dubey*

Research Scholar, Shri Krishna University, Chhatarpur M.P.