Statistical Methodology 33 (2016) 147–159

Contents lists available at ScienceDirect

Statistical Methodology

journal homepage: www.elsevier.com/locate/stamet

Discrete time software reliability modeling with periodic debugging schedule

Sudipta Das *, Anup Dewanji, Debasis Sengupta

Applied Statistical Unit, Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Article history:
Received 8 March 2016
Received in revised form
20 May 2016
Accepted 31 August 2016
Available online 13 September 2016

Keywords:
Discrete time scale
Periodic debugging
Profile likelihood
Covariance matrix
Coverage probability
Heterogeneous error detection probability

ABSTRACT

In many situations, multiple copies of a software are tested in parallel with different test cases as input, and the detected errors from a particular round of testing are debugged together. In this article, we discuss a discrete time model of software reliability for such a scenario of periodic debugging. We propose likelihood based inference of the model parameters, including the initial number of errors, under the assumption that all errors are equally likely to be detected. The proposed method is used to estimate the reliability of the software. We establish asymptotic normality of the estimated model parameters. The performance of the proposed method is evaluated through a simulation study and its use is illustrated through the analysis of a dataset obtained from testing of a real-time flight control software. We also consider a more general model, in which different errors have different probabilities of detection.

1. Introduction

Computer driven systems that pervade almost every sphere of modern life are dependent on their underlying software. Hence, the quality of the software influences the performance of the overall system. Software reliability is one of the most important parameters of judging the quality of a software. Depending on the use of modular structure, software reliability models can be divided into two classes: white-box and black-box [5]. A white-box model is used to estimate the reliability based on the specific modular structure of a given software. The black-box models view software as an unknown

* Corresponding author.
E-mail addresses: jusudipta@gmail.com (S. Das), dewanjia@isical.ac.in (A. Dewanji), sdebasis@isical.ac.in (D. Sengupta).

http://dx.doi.org/10.1016/j.stamet.2016.08.006
1572-3127/

unit ignoring the modular structure, although, they have often found to have fairly good capability of estimating reliability [1,11]. Starting with Jelinski and Moranda [9], a number of black-box models have been suggested for estimation of the reliability of a software (see [7] for a recent review). These models depend on failure data which are recorded continuously over time (e.g., calendar or execution time or staff hours).

In practice, however, there are many systems providing service over well-defined episodes of usage. For example, transactions in an ATM machine, smart card punching, etc., in which a successful completion of service with a given specification of usage is more important than the exact epoch of a particular failure of the software. The software used in this kind of systems does not operate continuously; rather, they are used intermittently for different sets of executions. Hence, the nature of its usage is truly discrete. A discrete time formulation is suitable for specification of reliability of such a software, and also for collecting data to estimate that reliability. Sometimes, even when a software is tested continuously, discretization of time occurs due to logistical issues of record-keeping. Discretely generated software fault data may also be available in situations, where the testing is done for the sake of improving the software, and assessing its reliability is not necessarily the primary objective. For this kind of software systems, one needs a different kind of modeling by considering each run of the software as one discrete unit of time and develop a model based on this discrete time scale, when inputs are selected from a suitable input space.

Different kinds of discrete time software testing models have been developed by Yamada and Osaki [16], Inoue and Yamada [8] and Kapur et al. [10], among others. However, these models assume that the errors are removed as and when they are detected. This assumption does not hold in many practical situations, when there are a few scheduled rounds of debugging, and the software testing goes on till the last debugging time. Specifically, the errors are not removed at the discrete time of detection; only a record of these detected errors is maintained. These are removed, with certainty, at the subsequently scheduled debugging time (see Worwa [15]; Yang and Chao [17]), perhaps by a team that is different from the debugging team. This type of debugging scheme is termed as 'periodic debugging schedule' in Dewanji et al. [4], where the authors consider a discrete time regression model, using a modular structure of the software, suitable for periodic debugging schedule. The periodic debugging set-up permits utilization of information from multiple detection of a single error. More importantly, it represents a prevalent mode of debugging in many software development environments, which has received surprisingly scant attention from reliability theorists.

In many models of software reliability, one makes the convenient assumption that all the errors contained in the software have the same severity. In reality, however, some errors have higher probability of appearance than other errors [14]. In our work, we consider analysis of data arising from discrete time periodic debugging scheme, without requiring any modular structure of the software, where probability of appearance of errors may or may not be equal. We estimate the initial number of errors, which determines the number of remaining errors in the software along with the parameters associated with our model for software reliability. We define reliability of the software as the probability of no failure in a complete run carried out with a randomly selected input. Generally, one may define reliability as the probability of no failure of the software in a fixed number of independent runs, which can be derived from the probability of no failure in a single run.

In Section 2, we present a discrete time software reliability model, assuming discrete time operation, homogeneous error probability and periodic debugging, and then construct the corresponding likelihood. In Section 3, we provide a computational method for estimating the initial number of errors. We discuss the asymptotic distribution of this estimator in Section 4. In Section 5, we report results of a simulation study to investigate the properties of the proposed estimator. We extend our analysis to a heterogeneous error probability model, and present a related simulation study, in Section 6. We illustrate both the estimation methods through the analysis of a real life dataset in Section 7. Some concluding remarks are given in Section 8.

2. Modeling and likelihood

We assume that there are initially ν (finite but unknown) errors in the software. If a tester runs a test case, the software may or may not fail. If it fails, then the tester reports one or more errors