**A study of software reliability growth model with testing coverage**



*A thesis submitted in the fulfilment of the requirements for the award of the degree of*

**Master of Technology**

**in**

**Computer Science**

**By**

**Basrah Yousuf**

**Under the supervision of**

**Dr.Javed Iqbal**

**PG. DEPARTMENT OF COMPUTER SCIENCE**

**UNIVERSITY OF KASHMIR, SRINAGAR**

**Table 0f Contents**

**Chapter 1 INTRODUCTION**

**Chapter 2 NHPP MODELS**

**Chapter 3 PROPOSED MODEL**

**Chapter 4 DATA ANALYSIS AND MODEL COMPARISON**

**Chapter 5 CONCLUSIONS**

**certificate**

**Abstract**

Modeling software reliability accurately and predicting its possible trends are essential for determining the overall reliability of the software. Testing is the key activity for determining and removing faults. This research proposes a software reliability model that considers not only error generation but also fault removal efficiency combined with testing coverage information based on a nonhomogeneous Poisson process (NHPP). Testing coverage measures quantify the degree of thoroughness of testing. It helps software developers to evaluate the quality of tested software and determine how much more additional testing is needed to improve the reliability of the software. For customers, this information estimates the confidence of using software product.

During the past four decades, many software reliability growth models (SRGMs) based on NHPP have been proposed to estimate the software reliability measures, most of which have the same following agreements: 1) it is a common phenomenon that during the testing phase, the fault detection rate always changes; 2) as a result of imperfect debugging, fault removal has been related to a fault reintroduction rate. But in practical software developing process, fault removal efficiency cannot always be perfect, i.e. the failures detected might not be removed completely and the original faults might still exist and new faults might be introduced meanwhile, which is referred to as imperfect debugging phenomenon. In this study, a model aiming to incorporate fault introduction rate, fault removal efficiency and testing coverage into software reliability evaluation is developed, using testing coverage to express the fault detection rate and using fault removal efficiency to consider the fault repair. We compare the performance of the proposed model with several existing NHPP SRGMs using three sets of real failure data. The results exhibit that the model can give a good fitting and predictive performance.

**Introduction**

Software reliability is defined as the probability that the system will perform its intended function under specified working conditions for a specified period of time. Software reliability growth models are mathematical functions that describe the error detection and the removal process. They help developers to know whether or not the code is suitable for use and how much more testing is required if it is not ready yet to release the software. It also provides the estimate of the number of faults that the users will encounter when operating the software. In order to balance between system stability and software testing debugging costs, software reliability growth models are widely used by software developers and testing staff. In fact, accurately modelling the software reliability and predicting its possible trends are essential in determining the optimal policy of system release. An NHPP (non homogenous Poisson process) is used to describe the time-dependent behaviour of the cumulative number of errors detected up to a certain testing time. The general model with NHPP applied is formulated based on the following assumptions:

1. The software errors testing/debugging process follows an NHPP.

2. The software fault intensity rate at any time is proportional to the number of remaining faults in the software at that time.

3. A debugging procedure takes place immediately when a software error is detected.

Generally, the software testing/debugging process is modelled as an error counting process. A counting process {N(t), t ≥0} is said to be an NHPP with intensity function where N(t) follows a Poisson distribution with mean function m(t); The mean value function m(t) is the expected number of errors detected within time (0, t). The conditional software reliability R(x/t) is defined as the probability that no error is detected within the time interval (t, t+x), given that an error occurred at time t (t≥ 0, x>0). Note that x is an operating time period according to some practical and managerial requirements.

**NHPP Models**

**Goel-Okumoto Model**

This model, first proposed by Goel and Okumoto, is one of the most popular NHPP model in the field of software reliability modelling. It is also called the exponential NHPP model. Considering failure detection as a Non homogeneous Poisson process with an exponentially decaying rate function, the mean value function is hypothesized in this model as

m(t)=a [ 1- exp(-bt)]; a>0 ,b>0

where a and b are positive constants.

a= expected number of faults

b= failure occurrence rate.

**Yamada Delayed S-Shaped Model**

The Yamada Delayed S-Shaped model is a modification of the non homogeneous Poisson process to obtain an S-shaped curve for the cumulative number of failures detected such that the failure rate initially increases and later exponentially) decays. It can be thought of as a generalized exponential model with failure rate first increasing and then decreasing. The software error detection process described by such an S-shaped curve can be regarded as a learning process because the testers skills will gradually improve as time progresses. The mean value function is

m(t)= a (1-(1+bt)exp(-bt)); b>0

where a and b are the expected total number of faults to be

detected eventually and failure occurrence rate respectively.

**Inflected S-Shaped Model (0hba)**

This model solves a technical problem in the Goel-Okumoto model. It was proposed by Ohba and its underlying concept is that the observed software reliability growth becomes S-shaped if faults in a program are mutually dependent, i.e., some faults are not detectable before some others are removed. The mean value function is

m(t)=a((1-exp(-bt))/((1+c exp(-bt)))); a>0,b>0,c>0

The parameter c is the inflection rate that indicates the ratio of the number of detectable faults to the total number of faults in the software, a is the expected total number of faults to be eventually detected, b is the fault detection rate, and is the inflection factor.

**Yamada and others**

In general it is considered to be unrealistic in software reliability modeling to assume that the faults detected software testing are perfectly removed without introducing new faults. Yamada et.al proposed software reliability assessment models with imperfect debugging by assuming that new faults are sometimes introduced when the faults originally latent in the software system are corrected and removed during the testing phase. It is assumed that the fault detection rate is proportional to the sum of the numbers of faults remaining originally in the system and faults introduced by imperfect debugging. This model is described by a non homogeneous Poisson process. The mean value function is given by

m(t) =((ab)/(alpha+b))(exp(-alpha t)-exp(-bt)); a,b,alpha >0

**pham zang model**

m(t)=(1/(1+betaexp(-bt)))((a+c)(1-exp(-bt))-((ab)/(b-alpha))(exp(-alphat)-exbt)));a,b,c,alpha,beta>0

where alpha and beta are positive constants.

**Software reliability modeling**

**Basic assumptions**

The proposed model is based on NHPP, which is utilized to describe the failure phenomenon during the testing phase. The counting process {*N*(*t*), *t* ≥ 0} of NHPP is shown as follows:

The mean value function *m*(*t*) is given as follows:



where *λ*(*u*) is the fault intensity function.

The proposed model is based on the following assumptions:

1. Software faults’ occurrence and removal follow NHPP.
2. The software failure rate at any time is a function of fault detection rate and the number of faults remaining in the software at that time. The fault detection rate can be expressed by ;

; *c*(*t*) is the percentage of the code that has been examined up to time *t*, *c*′(*t*) is the derivative of the testing coverage function.

1. When a software failure is detected, an immediate debugging starts, and either (a) the total number of faults is reduced by one with probability *p*, or (b) the total number of faults remains the same with probability 1-*p*.
2. During the fault repair process, whether the fault is removed completely or not, new faults are introduced with a probability constant *α*.

**Model development**

Let *c*(*t*) represent the percentage of the code that has been covered up to time *t*. Here c(t) refers to any kind of coverage, e.g. statement coverage, branch coverage, C-use coverage and P-use coverage etc. Obviously, *c*(*t*) is an increasing function of testing time *t*. Usually, it increases very fast from the beginning of software testing process as more test cases are executed to examine the software; after some certain time point, the testing coverage’s increasing rate becomes flat and less because less testing coverage happens to realize the residual fault detection . Thus, a concave or S-shaped function may be used to model the testing coverage function. Apparently, (1-*c*(*t*)) denotes the percentage of the code that has not been examined by test cases up to time *t*. The derivative of testing coverage function, *c*′(*t*), denotes the coverage rate. Thus, the function *c*′(*t*)/(1-*c*(*t*)) is recommended to be used to denote the fault detection rate , which has been taken as the common assumption by SRGMs considering testing coverage.

Based on the above assumptions, the mean value function considering both fault removal efficiency and testing coverage can be got by solving the following differential equation:

**(1)**

where *a*(*t*) represents the fault content function of the software, *β* (beta) is proportionality constant, *p* is the fault removal efficiency, which means *p*% percentage of detected faults can be removed successfully during the developing process, *m*(*t*) denotes the expected fault number detected up to time *t*, and *pm*(*t*) is the expected fault number that can be eliminated completely, so [*a*(*t*)-*pm*(*t*)] represents the expected remaining fault number presented in the software at time *t*. It should be noted that, when *β* = 1 and *p* ≠ 1, the proposed model has the same form as which is recommended in [. When *β* ≠ 1 and *p* = 1, we can get the same form recommended in . Existing models generally assume that *p* equals to 100% .

the total fault number function *a*(*t*), is a linear function of the expected fault number detected up to time *t*. That is,



where a denotes the initial fault number presented in the software system before testing starts and α > 0.

Substituting a(t) f into [Eq (1)](https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0181524" \l "pone.0181524.e004), and solving it in terms of the initial condition that at t = 0, m(t) = 0, we can obtain;

**(2)**

where c(0) refers to the testing coverage function when t = 0.

The software reliability function based on the NHPP is shown as follows:



### A new model with testing coverage

Substituting different testing coverage function c(t) into Equation (2), we can get different mean value function m(t) correspondingly. As, mentioned above the testing coverage is a non negative and non decreasing function of testing time t That is, the testing coverage function may show an S-shaped varying trend which is suitable to be described by a concave curve.

c(t)= 1-(1+bt)e-bt **(3)**

Substituting the value of the coverage function c(t) in the mean value function we get;

m(t)=a/p- alpha[1-(1/e-bt (1+bt)(alpha-p)beta]

It should be noted that both error generation and fault removal efficiency as well as testing coverage are all combined into the proposed model.

**Data analysis and model comparison**

we evaluate the effectiveness of the proposed model by using the different datasets and compare the proposed model with the other models, the Goel and Okumoto model ,Yamada delayed S-shaped model, Ohba inflection S-shaped model , Yamada exponential imperfect debugging model , and Pham–Zhang model .

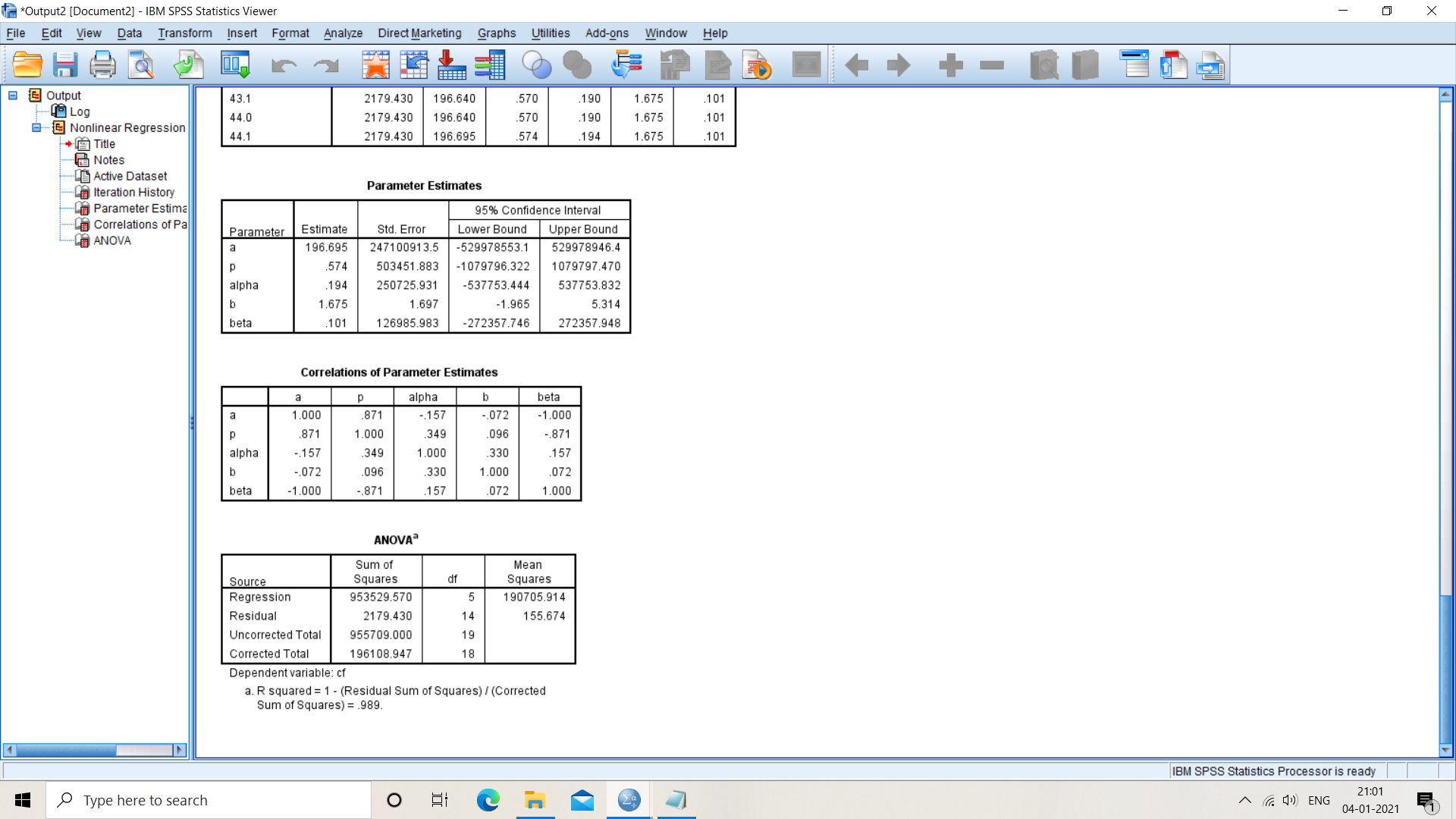
In order to investigate the effectiveness of the proposed model the comparison criteria chosen is R square measure.R square (Rsq) can measure how successful the fit is in explaining the variation of the data.

DATASET 1

PLI APPLICATION PROGRAM TEST DATA

There are total 328 faults detected within about 19 weeks. All data points are used to fit the models and estimate the models’ parameters.

|  |  |
| --- | --- |
| MODEL  GO MODEL  YAMADA S -SHAPED MODEL  INFLECTION MODEL  IMPERFECT DEBUGGING MODEL  PROPOSED MODEL | R SQUARE  0.986  0.984  0.992  0.987  **0.989** |



DATASET

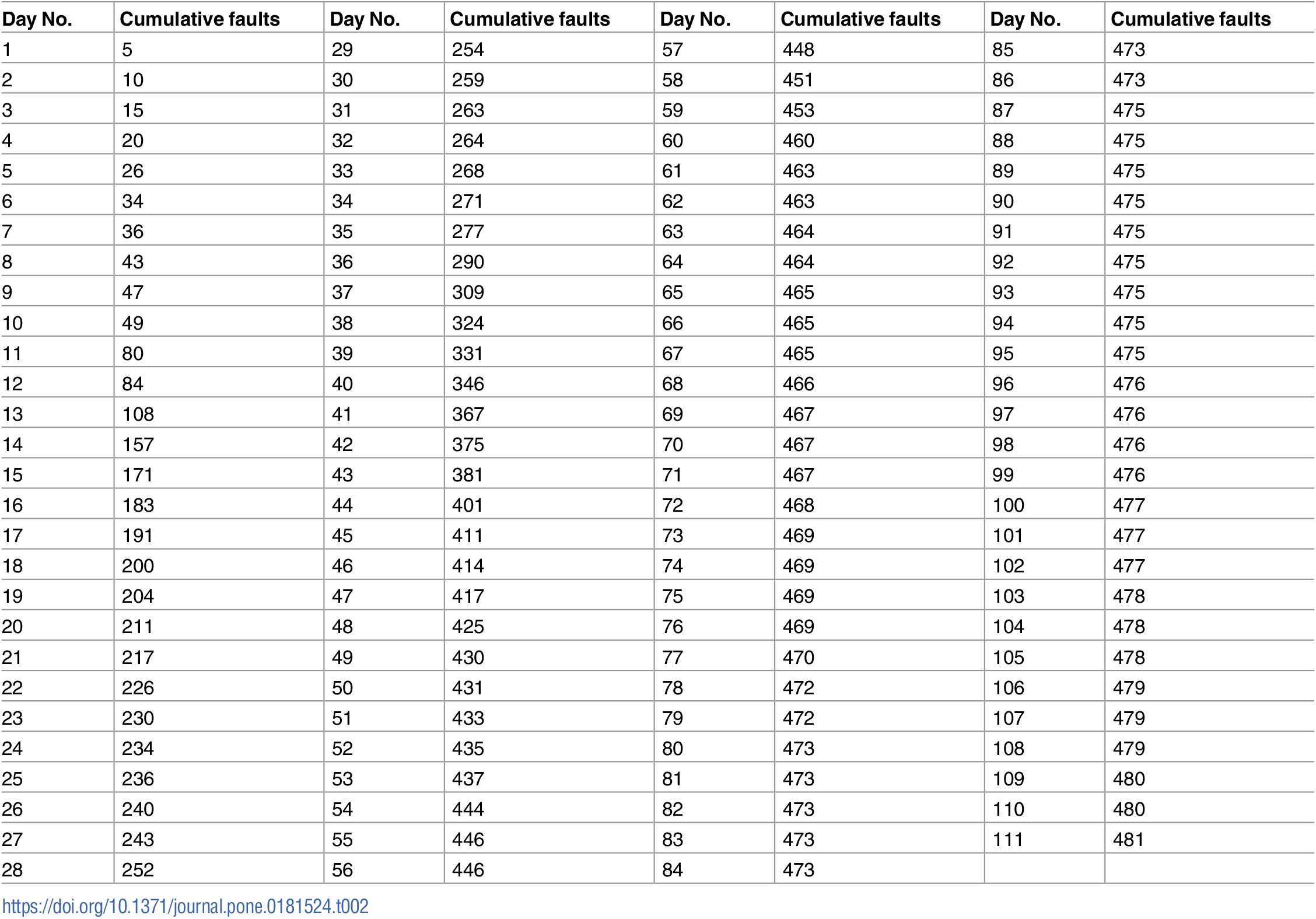
TANDEM COMPUTER DATA

we examine models using another data collected from Tandem Computers Release #2 , which has also been widely used in many studies. DS-2 is small in size and the failure data are tabulated in Table 4 with time unit week. There are totally 100 faults detected within about 20 weeks. All data points are used to fit the models and estimate the models’ parameters.

|  |  |
| --- | --- |
| MODEL  GO MODEL  YAMADA S -SHAPED MODEL  INFLECTION MODEL  IMPERFECT DEBUGGING MODEL  PROPOSED MODEL | R SQUARE  0.982  0.990  0.995  0.984  **0.991** |

### Monitor and control system data

The first data set is large in size and collected from testing a real monitor and control system (Data Set 1, DS-1) , which has been widely used in many studies. The details are recorded  and the time unit is day. All data points are used to fit the models and estimate the models’ parameters.



|  |  |
| --- | --- |
| MODEL  GO MODEL  YAMADA S -SHAPED MODEL  INFLECTION MODEL  IMPERFECT DEBUGGING MODEL  PROPOSED MODEL | R SQUARE  0.965  0.985  0.987  0.975  **0.985** |

### DATASET

### FAILURE DATA OF WIRELESS SYSTEM

|  |  |
| --- | --- |
| MODEL  GO MODEL  YAMADA S -SHAPED MODEL  INFLECTION MODEL  IMPERFECT DEBUGGING MODEL  PROPOSED MODEL | R SQUARE  0.986  0.990  0.994  0.986  **0.993** |

**Conclusions**

In this research, an imperfect debugging model NHPP-based is developed to incorporate both error generation and imperfect fault removal efficiency, together with considering the testing coverage to denote the fault detection rate function. Comparisons of this model with several other existing NHPP models have been presented in detail. In addition, various widely used failure data sets are provided for validating the goodness-of-fit and predictive performance of the proposed model. Thus, the results obtaining from the proposed model are encouraging.