https://orcid.org/0000-0001-7756-0004
https://orcid.org/0000-0001-8361-374X

UDC 620.169.1

**O.V. FEDUKHIN[*], A.A. MUKHA[*]**

# TO THE ISSUE OF PREDICTING THE NUMBER OF ERRORS IN THE OPERATION OF INFORMATION SYSTEMS SOFTWARE

[*]Institute of Mathematical Machines and Systems Problems of the National Academy of Sciences of Ukraine, Kyiv, Ukraine

***Анотація.*** *Оцінка надійності програмного забезпечення інформаційних систем є одним із найактуальніших питань сучасної галузі інформаційних технологій. Зважаючи на складність та важливість завдань, які виконуються інформаційними системами, особливо системами критичного призначення, питання забезпечення їхньої надійності стає все більш актуальним. У зв'язку з цим, вивчення та прогнозування залишкової кількості помилок, що можуть виникнути під час експлуатації програмного забезпечення, має надзвичайно велике значення. Ця стаття присвячена прогнозуванню залишкової кількості помилок проєктування у програмному забезпеченні інформаційних систем за результатами підконтрольної експлуатації. У роботі описано метод прогнозування кількості помилок проєктування, що базується на гіпотезі про випадковий марковський процес дифузійного типу з DN-розподілом напрацювання на помилку. Незважаючи на те, що цей розподіл традиційно використовувався як теоретична модель надійності елементів, пристроїв та систем обчислювальної техніки, він є дуже гнучкою функцією випадкового аргументу. Автори статті припускають, що такий розподіл доцільно випробувати як модель, що описує тренд процесу усунення накопичених при проєктуванні помилок програмного забезпечення, які призводять до його відмов. У результаті аналізу тематичних публікацій було сформовано контрольний приклад поведінки деякого програмного забезпечення в часі. Для цього прикладу відомі його реальні відмови за тривалий період експлуатації. Контрольні дані згодом порівнювалися з отриманими теоретичними результатами. Модель описано мовою Python, і обчислення проводилися у відповідному середовищі. У результаті моделювання отримано прогнозні дані про кількість відмов з використанням підходу на основі DN-розподілу напрацювання на помилку. Оцінка результатів проводилася за критерієм мінімального сумарного квадратичного відхилення.*

***Ключові слова:*** *програмне забезпечення, модель відмов, імовірнісно-фізичний підхід, DN-розподіл.*

***Abstract.*** *The assessment of software reliability in information systems is one of the most pressing issues in the modern IT industry. Considering the complexity and importance of tasks performed by information systems, especially systems of critical purpose, the necessity of ensuring their reliability is becoming more and more urgent. In this regard, studying and predicting the residual number of errors that may occur during the operation of the software is extremely important. This article is dedicated to forecasting the residual number of design errors in the software of information systems based on the results of the controlled operation. The paper describes a method for predicting the number of design errors, based on the hypothesis of a random Markov diffusion process with a DN-distribution for time between failures. Although this distribution has been traditionally used as a theoretical reliability model for components, devices, and computer systems, it is a very flexible function of a random argument. The authors of the article suggest that this distribution is worth testing as a model that describes the trend of eliminating accumulated design errors in software that lead to failures. As a result of analyzing thematic publications, a control example of the behavior of some software over time was formed. For this example, its actual failures over a long period of operation are known. The control data were subsequently compared with the obtained theoretical results. The model is described in Python, and calculations were carried out in the corresponding environment. As a result of the simulation, forecast data on the number of failures were obtained using the approach based on the DN-distribution for the time between failures. The evaluation of the results was assessed by the criterion of minimal sum of squared deviations.*

***Keywords:*** *software, failure model, probabilistic-physical approach, DN-distribution.*

© Fedukhin O.V., Mukha A.A., 2024

DOI: 10.34121/1028-9763-2024-3-4-124-131

## 1. Introduction

Currently, various information systems are widely used in everyday life, including systems for critical infrastructure. Alongside the benefits gained from their use, we face significant security risks on the one hand and substantial material losses due to downtime on the other. It should be unacceptable that reliability assessments are conducted only for the hardware or executive parts of systems designed for special and critical purposes. In this regard, the issues of software reliability and security are very relevant and require new, modern solutions.

Along with hardware defects and vulnerabilities, software failures are equally significant in the overall concept of ensuring the security of information systems. A large number of studies are dedicated to software reliability, where the primary research tool is the mathematical apparatus of probability theory and mathematical statistics, which serves as the foundation of the reliability theory of technical systems.

*The aim of the paper is* to develop a methodology for predicting software reliability using the *DN*-distribution and initial data on controlled operation at the early stages.

## 2. Highlighting unresolved issues

In work [1], a comparative analysis of software reliability models for one of the information systems is presented (Table 1). The study analyzes the degree of correspondence between the considered models and some experimental data obtained from the software operation results. To quantitatively assess the degree of correspondence of the models, the criterion of minimum total squared deviation (MTSD) was used $D_{total}$(2).

$$d_i^2 = (y_i - \hat{y}_i)^2. \tag{1}$$

$$D_{total} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2}$$

where $y_i$ is the observed value, $\hat{y}_i$ is the predicted value, $n$ is the number of values, and $d_i$ is the square deviation for the $n$ value of the series.

Table 1 — Comparative analysis of software reliability models

| Software operating time, hours | 800 | 1600 | 2400 | 3200 | 4000 | 4800 | 5600 | 6400 | 7200 | 8000 | 8800 | 9600 | Sum | MSTD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Experimental number of errors | 8 | 6 | 6 | 4 | 3 | 6 | 3 | 2 | 1 | 0 | 2 | 1 | 42 | 0 |
| Transient Process Model | 8.01 | 6.64 | 5.51 | 4.57 | 3.79 | 3.15 | 2.22 | 1,75 | 1.39 | 1.10 | 0.88 | 0.70 | 42 | 13.37 |
| Jelinski-Morande model | 8.07 | 6.65 | 5.48 | 4.51 | 3.73 | 3.06 | 2.20 | 1.73 | 1.37 | 1.09 | 0.87 | 0.69 | 41.33 | 17.14 |
| Schick-Wolverton model | 8.06 | 6.51 | 5.24 | 4.23 | 3.41 | 2.75 | 2.03 | 1.56 | 1.22 | 0.94 | 0.73 | 0.57 | 38.5 | 68.99 |

Continuation of Table 1

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Exponential model | 8.01 | 6.47 | 5.35 | 4.31 | 3.29 | 2.94 | 1.83 | 1.61 | 1.19 | 0.94 | 0.73 | 0.57 | 40.07 | 13.92 |
| Musa's Exponential Model | 7.95 | 7,52 | 5.34 | 3.93 | 3.06 | 2.42 | 2.01 | 1.29 | 1.05 | 0.87 | 0.73 | 0.60 | 39.96 | 13.91 |
| Morand's geometric model | 13.09 | 7.52 | 5.61 | 3,99 | 3.04 | 2.53 | 1.67 | 1.29 | 1,05 | 0.87 | 0.73 | 0.60 | 46,01 | 71.19 |
| Weibull model | 14.87 | 6.35 | 5.26 | 3.94 | 3/05 | 2.53 | 1,89 | 1.31 | 1,01 | 0.82 | 0,69 | 0.55 | 47.03 | 64.85 |
| Duane Model | 8.00 | 5.83 | 4.49 | 3.40 | 2.91 | 2.33 | 1.83 | 1,31 | 1,01 | 0.70 | 0.62 | 0.55 | 46.04 | 22.4 |
| Musa-Okumoto logarithmic model | 7.33 | 4.92 | 3.48 | 2.46 | 1.85 | 1.40 | 1.14 | 0.95 | 0.82 | 0.70 | 0.62 | 0.55 | 51.85 | 27.8 |
| S-shaped reliability growth model with kinks | 4.52 | 3.55 | 2.77 | 2,32 | 1.97 | 1.72 | 1.55 | 1.42 | 1.32 | 1.23 | 1.16 | 1.10 | 30.3 | 43.97 |

The analysis of the results showed that not all the considered models ensure the required accuracy in predicting the number of software errors. It is also noted that these models have diverse applications and are intended for modeling various classes of information systems. However, it should be noted that many of these models are based on the assumption that the occurrence of errors in software operation is considered a Markov random process, and the distribution of the random variable is represented by an exponential distribution with a constant failure rate.

It is an undisputed fact that during the operation of the software and the correction of emerging errors in the normal course (without introducing additional ones), the rate of occurrence of errors decreases over time. Consequently, error stream models based on the exponential distribution do not correspond to reality [2–4].

## 3. Description of the main research material

### 3.1. Selection of a theoretical model for software reliability

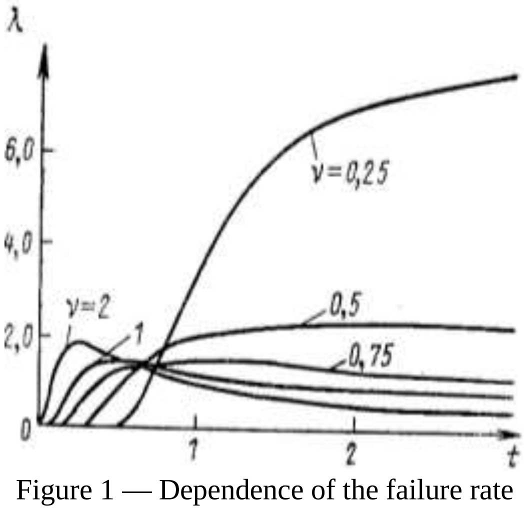

Figure 1 — Dependence of the failure rate of the *DN*-distribution on the shape $\nu$ parameter with the scale parameter $\mu = 1$

According to the authors, the reliability model based on the *DN*-distribution [5] can be widely and effectively used for predicting software failures. Examples of the behavior of the intensity function over time with fixed distribution law parameters are shown in Fig. 1.

This distribution is specifically formalized for a Markov random process of the diffusion type with non-monotonic realizations and constant velocity. The model parameters have a physical interpretation as the average speed of the random process and the coefficient of variation of the process realizations over time.

The type of distribution function $\lambda(t)$ plays a significant role in solving several reliability problems. This characteristic $\lambda(t)$ is particularly important when selecting a theoretical distribution function. The failure rate of the *DN*-distribution represents a non-monotonic function that starts from zero and reaches a final steady-state value $\frac{1}{2\,\mu\nu^2}$. A typical appearance of this family of curves for different values of the shape parameter, the $\nu$ coefficient of variation, is shown in Fig. 1. When applying this distribution in software reliability tasks, the most appropriate range for the coefficient of variation $\nu$ is from 0.5 to 1.0.

The function for the density distribution of time to failure is the following:

$$f(t) = \frac{1}{\nu t\sqrt{2\pi a t}} \exp\left[-\frac{(1-at)^2}{2\nu^2 at}\right]. \tag{3}$$

Density (3) corresponds to the distribution function $F(t)$:

$$F(t) = \Phi\left(\frac{at-1}{\nu\sqrt{at}}\right) + \exp\left(\frac{2}{\nu^2}\right)\Phi\left(-\frac{at+1}{\nu\sqrt{at}}\right), \tag{4}$$

where $\Phi(z) = \frac{1}{\sqrt{2\pi}}\int\limits_{-\infty}^{z}\exp\left(-\frac{x^2}{2}\right)dx$ is the normalized distribution.

In some cases, for the convenience of research, another designation of the scale parameter is adopted in the form $\mu$, the value of which is equal to the inverse value of the average rate of the degradation process $\left(\mu = \frac{1}{a}\right)$. The shape parameter $\nu$, as noted above, also represents the coefficient of variation of the degradation process. Let's write expressions (3), (4) in new parameters. The density of the *DN*-distribution is as follows:

$$f(t) = f_{DN}(t;\mu,\nu) = \frac{\sqrt{\mu}}{\nu t\sqrt{2\pi t}}\exp\left[-\frac{(t-\mu)^2}{2\nu^2\mu t}\right]. \tag{5}$$

The density (3) corresponds to the integral function of the *DN*-distribution:

$$F(t) = DN(t;\,\mu,\nu) = \Phi\left(\frac{t-\mu}{\nu\sqrt{\mu t}}\right) + \exp\left(\frac{2}{\nu^2}\right)\cdot\Phi\left(-\frac{t+\mu}{\nu\sqrt{\mu t}}\right). \tag{6}$$

The main distribution characteristics are given in Table 2.

Table 2 — *DN*-distribution characteristics

| Characteristic, designation | *DN*-distribution |
|---|---|
| Mathematical expectation, $M[T]$ | $\mu$ |
| Dispersion, $D[T]$ | $\mu^2\nu^2$ |
| Coefficient of variation, $\mathrm{V}[T]$ | $\nu$ |
| Probability of failure-free operation, $R(t)$ | $\Phi\left(\frac{\mu-t}{\nu\sqrt{\mu t}}\right) - \exp\left(\frac{2}{\nu^2}\right)\Phi\left(-\frac{\mu+t}{\nu\sqrt{\mu t}}\right)$ |
| Failure rate, $\lambda(t)$ | $\dfrac{\left(\nu t\sqrt{2\pi t}\right)^{-1}\sqrt{\mu}\,\exp\left[-\frac{(t-\mu)^2}{2\nu^2\mu t}\right]}{\Phi\left(\frac{\mu-t}{\nu\sqrt{\mu t}}\right) - \exp\left(\frac{2}{\nu^2}\right)\Phi\left(-\frac{\mu+t}{\nu\sqrt{\mu t}}\right)}$ |

### 3.2. Parameterization of the *DN*-distribution

Let's analyze the expressions for sample estimates of distribution parameters.
The calculation of a sample estimate of the coefficient of variation:

$$\nu = \frac{\sqrt{D}}{S}, \tag{7}$$

where $D$ is the sample variance of a random variable:

$$D = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2; \tag{8}$$

$S$ is the mean sample value of a random variable:

$$S = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{9}$$

The calculation of mean time to failure (MTTF):

$$\mu = \frac{\sum_{i=1}^{n} t_i}{\sum_{i=1}^{n} f_i}, \tag{10}$$

where $t_i$ is the time to detect an error, $f_i$ is the number of errors during the period.

Selective assessment of the average speed of the random error occurrence process looks like

$$a = \frac{1}{\mu}. \tag{11}$$

### 4. Software error prediction

The results of the controlled software operation in the form of a time series of software error detection are presented in Table 1.

The sample characteristics of the presented series of random events (model configuration) are as follows:

1. Let's calculate the variance of the random variable — the number of errors:

| Let's find the differences between each element and the mean value: | We raise each difference to the power of two: |
|---|---|
| $8 - 3.5 = 4.5$ | $(4.5)^2 = 20.25$ |
| $6 - 3.5 = 2.5$ | $(2.5)^2 = 6.25$ |
| $6 - 3.5 = 2.5$ | $(2.5)^2 = 6.25$ |
| $4 - 3.5 = 0.5$ | $(0.5)^2 = 0.25$ |
| $3 - 3.5 = -0.5$ | $(-0.5)^2 = 0.25$ |
| $6 - 3.5 = 2.5$ | $(2.5)^2 = 6.25$ |
| $3 - 3.5 = -0.5$ | $(-0.5)^2 = 0.25$ |
| $2 - 3.5 = -1.5$ | $(-1.5)^2 = 2.25$ |
| $1 - 3.5 = -2.5$ | $(-2.5)^2 = 6.25$ |
| $0 - 3.5 = -3.5$ | $(-3.5)^2 = 12.25$ |
| $2 - 3.5 = -1.5$ | $(-1.5)^2 = 2.25$ |
| $1 - 3.5 = -2.5$ | $(-2.5)^2 = 6.25$ |

$$D = \frac{20.25+6.25+6.25+0.25+0.25+6.25+0.25+2.25+6.25+12.25+2.25+6.25}{11} = 6.25.$$

2. Let's calculate the expected value of a random variable.

$$S = \frac{8+6+6+4+3+6+3+2+1+0+2+1}{12} = 3.5.$$

3. Let’s calculate the variation coefficient.

$$\upsilon = \frac{\sqrt{6{,}25}}{3{,}5} = 0.71.$$

The process of modeling is implemented using Python with the libraries NumPy and SciPy.

Step 1: For each time interval, calculate the mean time to failure (MTTF).

Step 2: Compute the probability of failure within the time interval:

$$P(t_0 < T \le t_1) = F(t_1, \upsilon, a) - F(t_0, \upsilon, a),$$

where $t_0$ is the initial time of interval, $t_1$ is the end time of interval, and $\mathrm{F}(\mathrm{t}, \upsilon, \mathrm{a})$ is the software error distribution function (4).

Step 3: Predict the number of errors $F_f$ .

Predicted number of errors at each time interval:

$$F_f = P(t_0 < T \le t_1) * F_0,$$

where $F_0$ is the number of errors identified at the beginning of forecasting.

The obtained forecasting results are shown in Fig. 2.

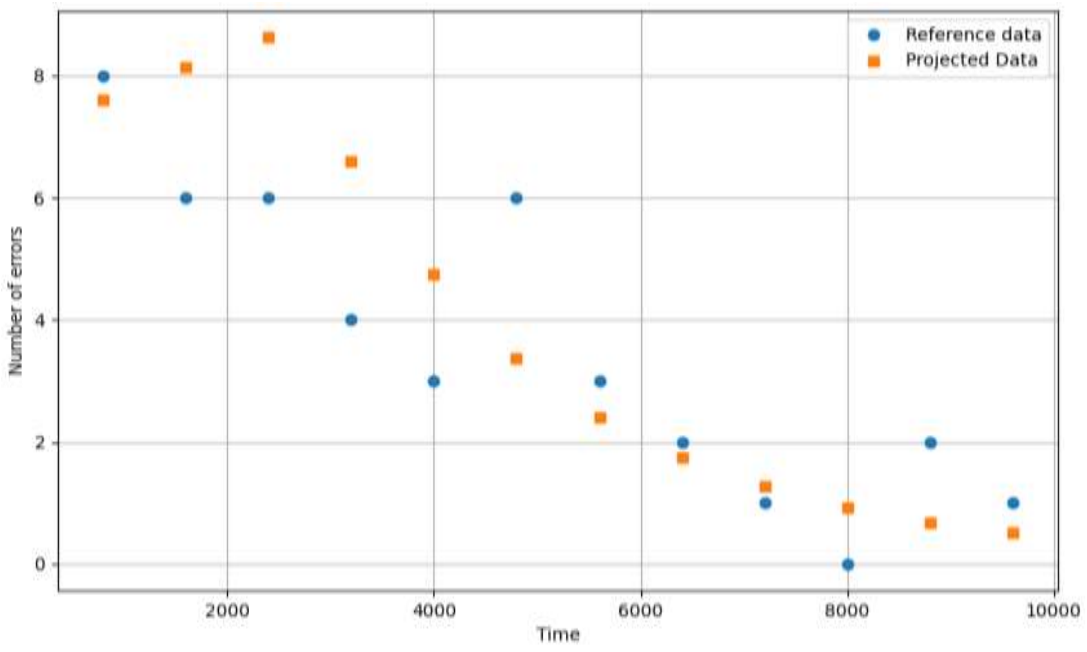

Figure 2 — Dynamics of the number of software errors over time

Let’s calculate the MTSD (Table 3).

Table 3 — Calculation of the MTSD

| i | $y_i$ | $\hat{y}_i$ | $d_i^2 = (y_i - \hat{y}_i)$ |
|---|---|---|---|
| 1 | 8 | 7.60 | $(8-7.60)^2 = 0.1600$ |
| 2 | 6 | 8.13 | $(6-8.13)^2 = 4.5449$ |
| 3 | 6 | 8.64 | $(6-8.64)^2 = 6.9696$ |
| 4 | 4 | 6.61 | $(4-6.61)^2 = 6.8121$ |
| 5 | 3 | 4.74 | $(3-4.74)^2 = 3.0276$ |
| 6 | 6 | 3.37 | $(6-3.37)^2 = 6.8913$ |
| 7 | 3 | 2.41 | $(3-2.41)^2 = 0.3481$ |
| 8 | 2 | 1.74 | $(2-1.74)^2 = 0.0676$ |
| 9 | 1 | 1.26 | $(1-1.26)^2 = 0.0676$ |
| 10 | 0 | 0.93 | $(0-0.93)^2 = 0.8649$ |
| 11 | 2 | 0.69 | $(2-0.69)^2 = 1.7161$ |
| 12 | 1 | 0.51 | $(1-0.51)^2 = 0.2401$ |
| Sum | | | 31.71 |

Based on the results of calculating the MTSD, it is possible to compare different software models with each other (Table 4).

Table 4 — Ranking of models according to the MTSD criteria

| Rank | Model | MSTD |
|---|---|---|
| 1 | Transition process model | 13.37 |
| 2 | Exponential Musa model | 13.91 |
| 3 | Simple exponential model | 13.93 |
| 4 | Jelinsky-Moranda model | 17.15 |
| 5 | Duena model | 22.40 |
| 6 | Logarithmic Musa-Okumoto model | 27.80 |
| 7 | **Diffusion-non-monotonic model** | **31.71** |
| 8 | S-shaped growth of reliabilitywith kinks | 43.97 |
| 9 | S-shaped growth of reliability | 57.02 |
| 10 | Weibull model | 64.86 |
| 11 | Schick-Wolverton model | 69.00 |
| 12 | Geometric Morandi model | 71.20 |

## 5. Conclusions

As a result of the analysis of the obtained data, it can be stated that the calculated RMSE of 37.71 indicates that the diffusion-non-monotonic model is ranked in the middle of the list in terms of forecasting accuracy. This result can be considered satisfactory, given the relative simplicity of this method.

The reason for the average accuracy of the forecast using the proposed diffusion-non-monotonic model lies in the fact that its formalization [3] is based on the hypothesis of a constant average degradation rate over time. In reality, however, based on general considerations and experiments, the average rate decreases along a curve close to an exponential one, but only if the detected errors are corrected without introducing new design errors. Otherwise, the adequacy of the forecast using the diffusion-non-monotonic model will be significantly higher than that of the models based on an exponential dependency, due to the assumption of the non-monotonicity of the realizations of the random process.

A major advantage of the method is that in order to start modeling, it is sufficient to have data on only one error detected at the initial stage of testing or operation. If there is a prior estimate of the variation coefficient of the process of detecting and correcting design errors, forecasting using the diffusion-non-monotonic model is possible even in the absence of software error statistics, which other competing models do not allow.

As a promising direction for the development of this approach, it can be noted that the accuracy of the forecasting method can be significantly improved by introducing additional parameters of the random process, such as the defect detection rate, the defect correction rate, etc.

## REFERENCES

1. Маевский Д.А., Яремчук С.А. Анализ моделей надежности программного обеспечения гарантоспособных информационных систем. *Електромашинобудування та електрообладнання*. 2010. № 76. С. 68–79.

2. Ivanov N.N. A probabilistic model of diffusion distribution for reliability assessment of electronic devices. *Information and Control Systems.* 2012. N 4 (59). P. 65–68.

3. Emelyanov V.S., Rabchun A.V. Region of applicability of diffusion distributions in reliability problems. *Atomic Energy*. 1991. Vol. 71. P. 552–556.

4. Hasilova K., Vališ D. Composite laminates reliability assessment using diffusion process backed up by perspective forms of non-parametric kernel estimators. *Engineering Failure Analysis*. 2022. Vol. 138. P. 128.

5. Strelnikov V.P., Fedukhin O.V. Assessment and forecasting of reliability of electronic components and systems. K.: Logos, 2002. 486 p.

*Стаття надійшла до редакції 13.08.2024*