International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-2, December 2019

Efficient and Convenient Application to Determine the Functions and Analysis of the Reliability of the Device

Aslonov Qodir Ziyodullayevich, Zhalilov Rashid Babakulovich, Mukhamadieva Zarina Bakhodirovna, Safarov Alisher Bekmurodovich, Mamedov Rasul Akif-ogli

Abstract; The article describes the types of distribution laws that can be used to determine the reliability of the device (element), analyzed the most frequently used methods for calculating the reliability. When calculating the reliability functions, the modern programming language "Swift" is used and their algorithms are described.

Key words: probability of uptime, probability of failure, average uptime, failure rate.

I. INTRODUCTION

In today's modern society industrial production plays huge role in satisfying material and intellectual demands, and obviously it in itself leads to expansion of the production volume. As a result, increasing and investigating reliability of the technologies and system being utilized becomes a major problem [1]. The efficient estimate of failure probability with accuracy is the main topic in reliability analysis [3]. To assess major features and settings of reliability of the technical units and their composition can be applied a number of different laws of distribution [1].

One of the major settings of reliability is probability of uptime of the device p(t), t is index of time. The formula of probability of uptime is as following [1]:

$$p(t) = P\{\xi > t\}, \qquad t \geq 0 \tag{1}$$

If put forward hypothesis that the device is operable: $p(0) = 1$. As time passes probability of operable capability decreases. Because with lapse of time every possible device losses working-capacity.

Another setting of reliability, this is probability of failure of the device q(t).

Revised Manuscript Received on December 30, 2019.
***** Correspondence Author

Aslonov Qodir Ziyodullayevich*, Ph.D. Studentof Information communications technology Department, Bukhara Engineering Technological Institute, Bukhara, Uzbekistan

Zhalilov Rashid Babakulovich, Associate Professor at the Department ofEnergy, Bukhara Engineering Technological Institute, Bukhara, Uzbekistan

Mukhamadieva Zarina Bakhodirovna, Ph.D. Studentof Information communications technology Department, Bukhara Engineering Technological Institute, Bukhara, Uzbekistan

Safarov Alisher Bekmurodovich, Ph.D. Studentof Energy auditDepartment, Bukhara Engineering Technological Institute, Bukhara, Uzbekistan

Mamedov Rasul Akif-ogli, Ph.D. Studentof Energy department, Bukhara Engineering Technological Institute, Bukhara, Uzbekistan

The formula probability of failure is as following:

$$q(t) = P\{\xi \leq t\} = 1 - p(t), \quad t \geq 0 \tag{2}$$

Another setting of reliability, this is distribution density of time of failure of the device $\omega(t)$.

The formula probability of failure is as following: density of distribution of time to failure

$$\omega(t) = \frac{dq(t)}{dt} = -\frac{dp(t)}{dt} \tag{3}$$

From (3) it follows that the probability of failure-free operation on the interval (0, t) is equal to the integral of the distribution density function from time t to ∞:

$$p(t) = \int_t^{\infty} \omega(t)dt$$

based on the formulas (2) and (3), we obtain the following:

$$p(t) = 1 - \int_0^t \omega(t)dt \tag{4}$$

Another setting of reliability, this is average uptime of the device T & failure rate (t). The formula average uptime is as following:

$$T = \int_0^{\infty} p(t)dt \tag{5}$$

and formula failure rate is as follows:

$$\lambda(t) = \frac{\omega(t)}{p(t)} \tag{6}$$

The failure of elements, as random events, is subject to various laws of distribution.

The main distribution laws used in the theory of reliability [6] are the Weibull distribution, the Exponential distribution, the Rayleigh distribution, and the normal distribution [1].

II. APPLICATION ARCHITECTURE

Desktop applications are based on design patterns or architectural schemes. In this type of application, the most important architectural software decisions are [2]: mathematical model of the program being created, development environment and application logic.

Hereafter, those issues are briefly addressed [2].

A. Mathematical model

The Weibull distribution**.** The Weibull distribution which was proposed by Waloddi Weibull in 1939 is a very important lifetime distribution and is widely used in many fields. However, the hazard function of the traditional Weibull distribution can only be increasing, decreasing or constant. To meet the need of fitting complex failure modes and the bathtub-shaped hazard rate, researchers have proposed many improved models based on the traditional Weibull distribution [4].

In the Weibull distribution, the probability of failure-free operation on the interval (0, t) is [1]:

$$p(t) = \exp\left(-\lambda t^{\delta}\right), t \geq 0;\ \lambda > 0;\quad \delta > 0 \tag{7}$$

and the distribution density of time to failure is equal to [1]:

$$\omega(t) = p'(t) = \lambda\delta t^{\delta-1}\exp\left(-\lambda t^{\delta}\right) \tag{8}$$

The formula average uptime is as following:

$$T = \int_0^{\infty} e^{-\lambda t^{\delta}} dt \ = \ \lambda^{\frac{1}{\delta}}\Gamma\left(1+\frac{1}{\delta}\right) \tag{9}$$

and formula failure rate is as follows [1]:

$$\lambda(t) = \frac{\omega(t)}{p(t)} = \frac{\lambda\sigma t^{\delta-1}\exp\left(-\lambda t^{\delta}\right)}{\exp(-\lambda t^{\delta})} = \ \lambda\delta t^{\delta-1},$$
$$t \geq 0; \lambda > 0; \delta > 0 \tag{10}$$

*The Exponential distribution***.** In determining reliability, the exponential distribution is also used. This distribution is considered to a part of Weibull distribution. Because, here $\delta = 1$ is considered. As a result, the probability of failure-free operation of device is equal to [1]:

$$p(t) = e^{-\lambda t}, t \geq 0;\ \lambda > 0 \tag{11}$$

The formula average uptime is as following [1]:

$$T = \int_0^{\infty} e^{-\lambda t} dt \ = \ \frac{1}{\lambda} \tag{12}$$

$$p(t) = e^{-\frac{1}{T}}, t \geq 0;\ T > 0 \tag{13}$$

The distribution density of time to failure is equal to [1]:

$$\omega(t) = p'(t) = \ \lambda e^{-\lambda t} \tag{14}$$

It should be noted that the duration of the period of normal operation before the onset of aging may be less than the average uptime. That's why it is necessary to consider, a span of time in which the exponential model can be used, it can be even less than average uptime.

*The Rayleigh distribution***.** In the Rayleigh distribution, the probability of failure-free operation on the interval (0, t) is equal to:

$$p(t) = exp\left(-\frac{t^2}{2\sigma^2}\right) \tag{15}$$

where: σ is the Rayleigh distribution parameter, which is simultaneously is a fashion of this distribution.

The distribution density of time to failure is equal to [1]:

$$\omega(t) = -p'(t) = \frac{t}{\sigma^2}\ exp\left(-\frac{t^2}{2\sigma^2}\right) \tag{16}$$

and formula failure rate is as follows:

$$\lambda(t) = \frac{\omega(t)}{p(t)} = \frac{t}{\sigma^2} \tag{17}$$

The formula average uptime is as following:

$$T = \int_0^{\infty} t\omega(t)dt = \int_0^{\infty} \frac{t^2}{\sigma^2} e^{-\frac{t^2}{\sigma^2}} dt = \sqrt{\frac{\pi}{2}}\sigma = 1.25\sigma \tag{18}$$

*The Normal distribution***.** The most commonly used distribution in statistics is the normal Gaussian distribution [10]. The distribution density of time to failure is equal to [1]:

$$\omega(t) = \frac{1}{\sqrt{2\pi\sigma^2}}\ exp\left(-\frac{(t-T)^2}{2\sigma_T^2}\right) \tag{19}$$

The reliability function in this case is equal to:

$$p(t) = \int_t^{\infty} \omega(t)dT = \int_t^{\infty} \frac{1}{\sqrt{2\pi\sigma_T^2}} e^{-\frac{(t-T)^2}{2\sigma_T^2}} dT =$$
$$\frac{1}{\sqrt{2\pi}} \int_{\frac{t-T_{o'r}}{\sigma_T}}^{\infty} e^{-\frac{1}{2}\left(\frac{t-T_{o'r}}{\sigma_T}\right)^2} d\left(\frac{t-T_{o'r}}{\sigma_T}\right) =$$
$$= \frac{1}{\sqrt{2\pi}} \int_{\frac{t-T_{o'r}}{\sigma_T}}^{\infty} e^{-\frac{x^2}{2}} dx \tag{20}$$

and formula failure rate is as follows [1]:

$$\lambda(t) = \frac{\omega(t)}{p(t)} = \frac{1}{\sqrt{2\pi\sigma_T^2}} exp\left\{-\frac{(t-T_{o'r})^2}{\sigma_T}\right\} \cdot \left[1 - F\left(\frac{t-T_{o'r}}{\sigma_T}\right)\right]^{-1} \tag{21}$$

B. Development environment

Software has to be stable and be characterized for its facility of managing and its coupling with the operating system that hostess it [4]. Programming language Swift, developed specially for MAC OS X operating system from Apple computers covers all these requirements [4]. This programming language is delivered together with the programming platform Xcode within the OS X operating system [4] (Fig. 1).

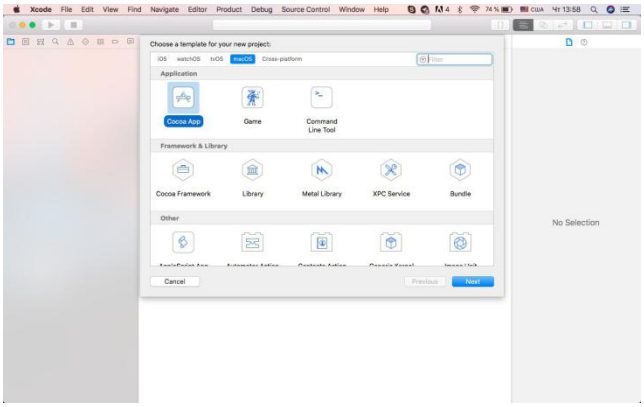

Fig. 1. Interface Xcode.

Xcode contains all the instruments needed by developers to build great quality products. The main languages used with it are Swift, Objective-C, C and C++ [8]. Tightly integrated with the Cocoa and Cocoa Touch frameworks, Xcode is an incredibly productive environment for building apps for Mac, iPhone, iPad, Apple Watch, and Apple TV [9].

C. Application logic

The model-view-controller (MVC) pattern (Fig. 2) is a classic procedure to develop interactive desktop applications [2]. It relies on a clean separation of objects into one of three categories: models for maintaining data, views for displaying all or a portion of the data, and controllers for handling events that affect the model or view [2].

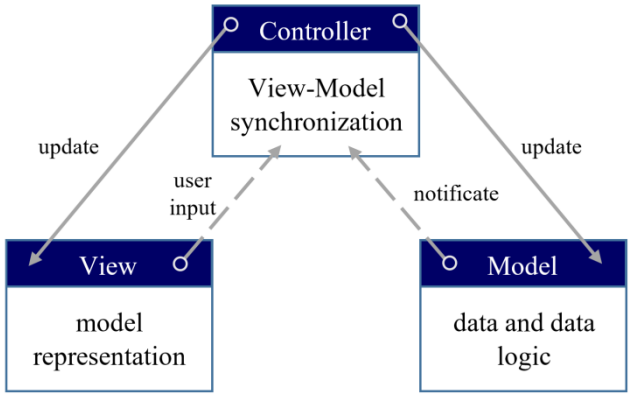

Fig. 2. Model-view-controller pattern [2].

This application is developed in the Swift programming language.

Swift is the result of the latest research on programming languages, combined with decades of experience building Apple platforms [7]. Named parameters are expressed in a clean syntax that makes APIs in Swift even easier to read and maintain. Even better, you don't even need to type semi-colons. Inferred types make code cleaner and less prone to mistakes, while modules eliminate headers and provide namespaces [7].

This programming language has the following achievements:

- To best support international languages and emoji, Strings are Unicode-correct and use a UTF-8 based encoding to optimize performance for a wide-variety of use cases [7].
- Memory is managed automatically using tight, deterministic reference counting, keeping memory usage to a minimum without the overhead of garbage collection [7].
- Swift eliminates entire classes of unsafe code. Variables are always initialized before use, arrays and integers are checked for overflow, memory is automatically managed, and enforcement of exclusive access to memory guards against many programming mistakes [7].
- Syntax is tuned to make it easy to define your intent - for example, simple three-character keywords define a variable ***var*** or constant ***let*** [7].
- And Swift heavily leverages value types, especially for commonly used types like Arrays and Dictionaries. This means that when you make a copy of something with that type, you know it won't be modified elsewhere [7].
- Another safety feature is that by default Swift objects can never be ***nil***. In fact, the Swift compiler will stop from trying to make or use a ***nil*** object with a compile-time error. This makes writing code much cleaner and safer, and prevents a huge category of runtime crashes in your apps.
- However, there are cases where ***nil*** is valid and appropriate. For these situations Swift has an innovative feature known as optionals. An optional may contain ***nil***, but Swift syntax forces you to safely deal with it using thesyntax to indicate to the compiler you understand the behavior and will handle it safely [7].

III. APPLICATION IMPLEMENTATION

The program interface consists of 5 views, and the structure of the application is as follows (Fig. 3):

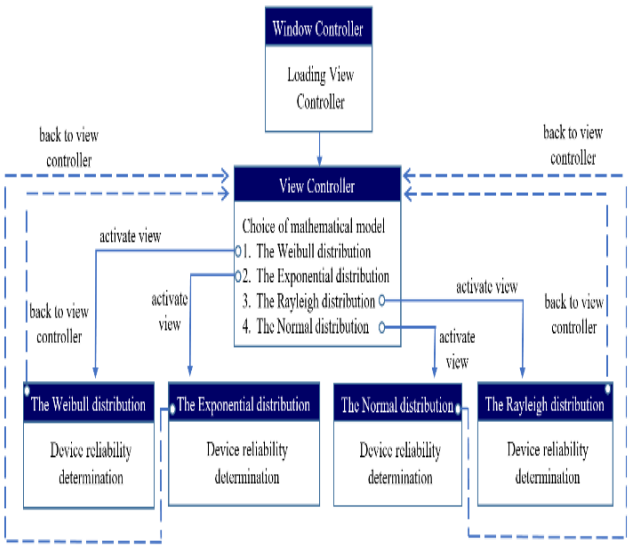

Fig. 3. Application structure.

In the first view, the user selects the distribution law (Fig. 4).

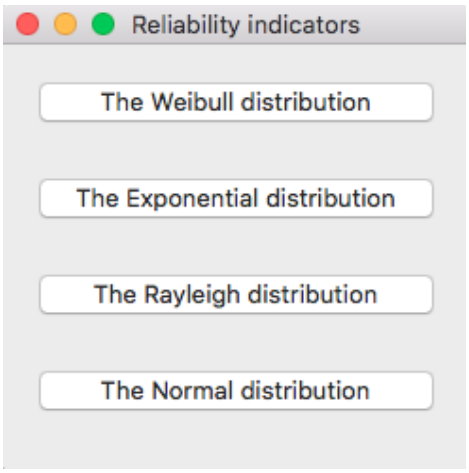

Fig. 4. Distribution law selection view.

Depending on the user's choice, the window is loaded into RAM.

A. The Weibull distribution view

Fig. 5 shows the program window, which determines the reliability of the device based on the Weibull distribution law.

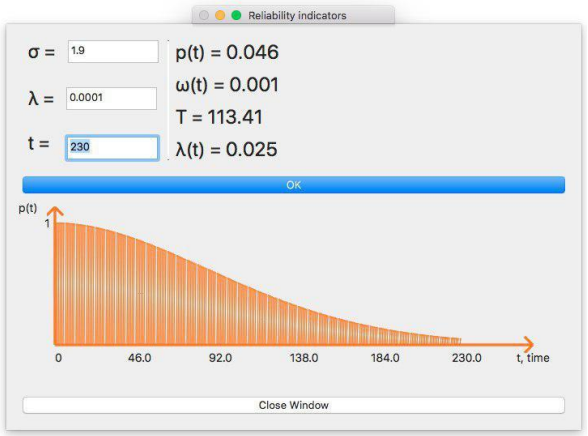

Fig. 5. The Weibull distribution view (interface).

This user interface consists of three parts (Fig. 6):

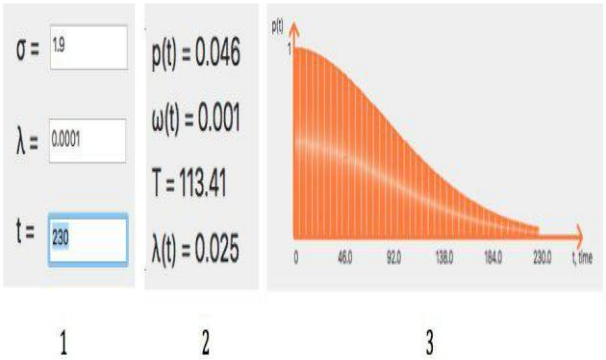

Fig. 6. The Weibull distribution view (interface).

1. Data input.
2. Reliability function definition. Fig. 7 shows a model for the implementation of the Weibull distribution.
3. Reliability change schedule.

```
1   import Cocoa
2   class Weibull: NSViewController {
        @IBOutlet weak var sigma:NSTextField!
        @IBOutlet weak var lyambda:NSTextField!
        @IBOutlet weak var vaqt:NSTextField!
        @IBOutlet weak var ptLabel:NSTextField!
        @IBOutlet weak var omegaTLabel:NSTextField!
        @IBOutlet weak var tsrLabel:NSTextField!
        @IBOutlet weak var lyambdatLabel:NSTextField!
        @IBOutlet weak var label1:NSTextField!
        @IBOutlet weak var label2:NSTextField!
        @IBOutlet weak var label3:NSTextField!
        @IBOutlet weak var label4:NSTextField!
        @IBOutlet weak var label5:NSTextField!
        @IBOutlet weak var label6:NSTextField!
        @IBOutlet weak var pt1label:NSTextField!
        @IBOutlet weak var vaqtlabel:NSTextField!
        @IBOutlet weak var birlabel:NSTextField!
19
20      override func viewDidLoad() { ••• }
33
        @IBAction func closeWindow(sender:NSButton){ ••• }
41
        @IBAction func weibullMethod(sender:NSButton){ ••• }
45
46      func natija(){ ••• }
121
122     func allert()->Bool { ••• }
130
131     func gammaFunction(number:Float)->Float80{ ••• }
169
170     func rounding(number:Float)->Int{ ••• }
215 }
216
217 extension Weibull: NSControlTextEditingDelegate { ••• }
```

Fig. 7 Swift file for model for the implementation of the Weibull distribution.

B. The Exponential distributionview

Fig. 8 shows the program window, which determines the reliability of the device based on the Exponential distribution law.

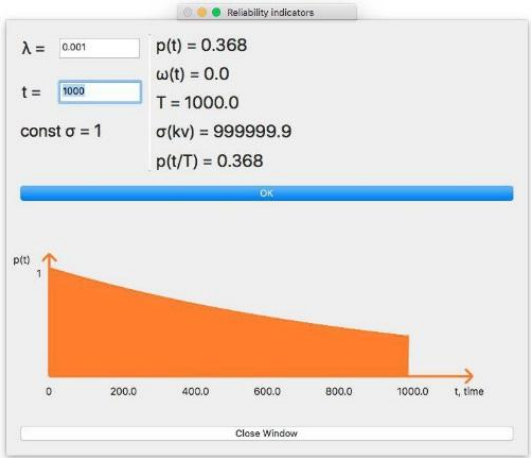

Fig. 8. The Exponential distribution view (interface).

This user interface consists of three parts (Fig. 9):

Fig. 9. The Exponential distribution view (interface).

1.Data input.
2.Reliability function definition. Fig. 10 shows a model for the implementation of the Exponential distribution.
3. Reliability change schedule.

```
import Cocoa
class ExponentMethod: NSViewController {
    @IBOutlet weak var lyambda:NSTextField!
    @IBOutlet weak var vaqt:NSTextField!
    @IBOutlet weak var ptLabel:NSTextField!
    @IBOutlet weak var omegaTLabel:NSTextField!
    @IBOutlet weak var tsrLabel:NSTextField!
    @IBOutlet weak var lyambdatLabel:NSTextField!
    @IBOutlet weak var ptTLabel:NSTextField!
    @IBOutlet weak var label1:NSTextField!
    @IBOutlet weak var label2:NSTextField!
    @IBOutlet weak var label3:NSTextField!
    @IBOutlet weak var label4:NSTextField!
    @IBOutlet weak var label5:NSTextField!
    @IBOutlet weak var label6:NSTextField!
    @IBOutlet weak var pt1label:NSTextField!
    @IBOutlet weak var vaqtlabel:NSTextField!
    @IBOutlet weak var birlabel:NSTextField!
    override func viewDidLoad() { ••• }

    @IBAction func closeWindow(sender:NSButton){ ••• }

    @IBAction func expMethod(sender:NSButton){ ••• }

    func natija(){ ••• }

    func allert()->Bool { ••• }

}

extension ExponentMethod: NSControlTextEditingDelegate {
    func controlTextDidEndEditing(_ notification: Notification) { ••• }
}
```

Fig. 10 Swift file for model for the implementation of the Exponential distribution.

C. The Rayleigh distributionview

Fig. 11 shows the program window, which determines the reliability of the device based on the Rayleigh distribution law.

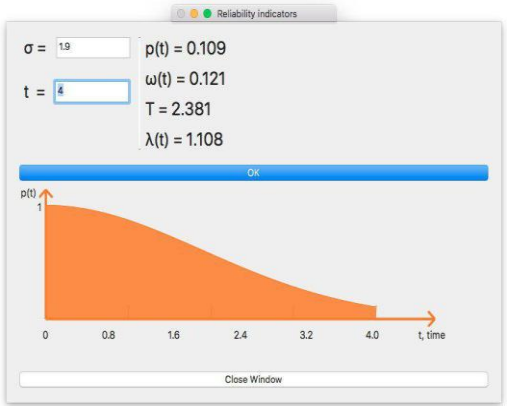

Fig. 11. The Rayleigh distribution view (interface).

This user interface consists of three parts (Fig. 12):

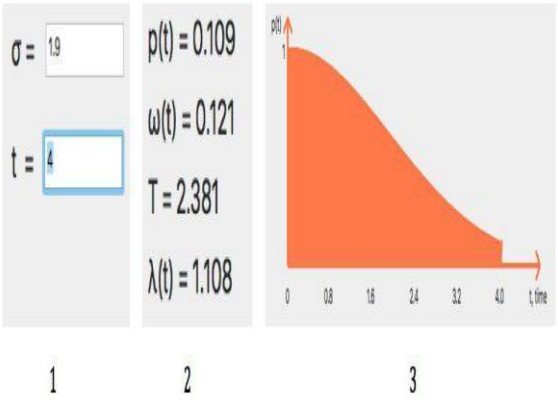

Fig. 12. The Rayleigh distribution view (interface).

1.Data input.
2. Reliability function definition. Fig. 13 shows a model for the implementation of the Rayleigh distribution.
3.Reliability change schedule.

```
import Cocoa
class RayleighMethod: NSViewController {
    @IBOutlet weak var sigma:NSTextField!
    @IBOutlet weak var vaqt:NSTextField!
    @IBOutlet weak var ptLabel:NSTextField!
    @IBOutlet weak var omegaTLabel:NSTextField!
    @IBOutlet weak var tsrLabel:NSTextField!
    @IBOutlet weak var sigmaLabel:NSTextField!
    @IBOutlet weak var label1:NSTextField!
    @IBOutlet weak var label2:NSTextField!
    @IBOutlet weak var label3:NSTextField!
    @IBOutlet weak var label4:NSTextField!
    @IBOutlet weak var label5:NSTextField!
    @IBOutlet weak var label6:NSTextField!
    @IBOutlet weak var pt1label:NSTextField!
    @IBOutlet weak var vaqtlabel:NSTextField!
    @IBOutlet weak var birlabel:NSTextField!
    override func viewDidLoad() { ••• }
    @IBAction func closeWindow(sender:NSButton){ ••• }

    @IBAction func rayleighMethod(sender:NSButton){ ••• }

    func natija(){ ••• }

    func allert()->Bool { ••• }
}

extension RayleighMethod: NSControlTextEditingDelegate {
    func controlTextDidEndEditing(_ notification: Notification) { ••• }
}
```

Fig. 13 Swift file for model for the implementation of the Rayleigh distribution.

D. The Normal distributionview

Fig. 14 shows the program window, which determines the reliability of the device based on the Normal distribution law.

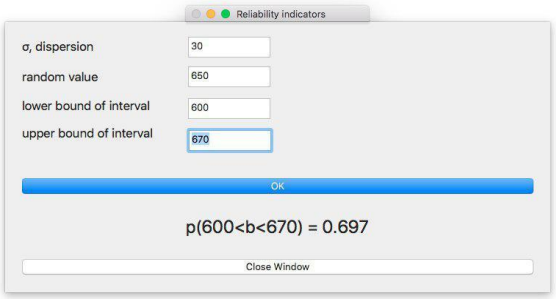

Fig. 14. The Normal distribution view (interface).

This user interface consists of two parts. (Fig. 15):

Fig. 15. The Normal distribution view (interface).

1.Data input.
2.Reliability function definition. Fig. 16 shows a model for the implementation of the Normal distribution.

```
import Cocoa
class NormalMethod: NSViewController {
    @IBOutlet weak var dispersiya:NSTextField!
    @IBOutlet weak var randomSize:NSTextField!
    @IBOutlet weak var startingValue:NSTextField!
    @IBOutlet weak var finalValue:NSTextField!
    @IBOutlet weak var pt1label:NSTextField!

    override func viewDidLoad() { ••• }

    @IBAction func closeWindow(sender:NSButton){ ••• }

    @IBAction func normalMethod(sender:NSButton){ ••• }
}
```

Fig. 16 Swift file for model for the implementation of the normal distribution.

IV. EVALUATION

When applying any law of reliability distribution in the program, the user receives not only the resulting values of the reliability function, but also the graph of the probability of the device operation. This graph reflects the operation of the device, also in what conditions it workswith full power and when low power. Based on this data, the user becomes familiar with the potential capabilities of the device and draws certain conclusions.

Task 1: Determine the average uptime T and failure rate λ(t) for the device, time of uptime the failure of obeys the Weibull law with parameters σ = 1.6, λ = 0.0001 1/hour during operation t = 317 hours.

The user enters three values:

1.Asymmetry: σ = 1.6

2.Intensity:λ = 0.0001 1/hours.

3.Time: t = 317 hours.

After that, the user presses the "OK" button and sees the reliability functions. Fig. 17 shows the results of this task.

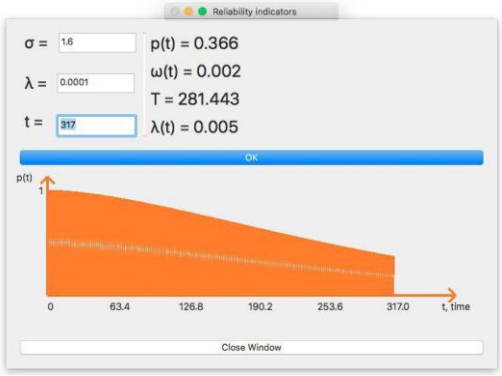

Fig. 17 Results of task 1.

If we change the time t = 510, then we get the following values (Fig. 18).

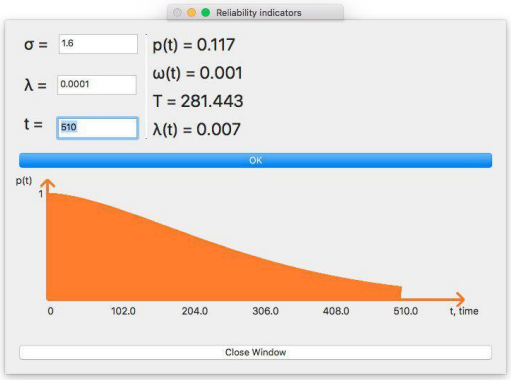

Fig. 18 Results of task 1 with changing time.

As shown in Figures 17 and 18, the user has the opportunity to evaluate the reliability of the device using graphic data.

V. CONCLUSIONS

This paper is a handy application for determining the reliability function of a device. With this application is determined not only the reliability functions, but also analyzes the probability of failure-free operation of the device. The application can be applied to students studying technical areas of higher education. The software is fully developed on the Xcode software environment and in the Swift programming language.

This software is far from being complete. In the future, client-server technologies and additions of the API (application programming interface[5]) module to the application will be considered.

REFERENCES

1. V.R. Matveyevsky. Reliability of technical systems. 2002. URL https://www.studm-ed.ru/matveevskiy-vr-nadezhnost-tehnicheskih-sistem_9e0b96a8add.html
2. J. Gracia, E. Bayo. An effective and user-friendly web application for the collaborative analysis of steel joints.Advances in Engineering Software 2018;119:60-67. https://doi.org/10.1016/j.advengsoft.2018.02.007.
3. Jun Xu, Shengyang Zhu. An efficient approach for high-dimensional structural reliabilityanalysis. Mechanical Systems and Signal Processing 2019;122:152-70. https://doi.org/10.1016/j.ymssp.2018.12.007.
4. Moumtadi Fatima, La Rotta Santos Pedro Felipe, Delgado Hernández Julio Carlos. Procedia Engineering 2012; 35:165-75. https://doi.org/10.1016/j.proeng.2012.04.177.
5. Congying Xu, Xiaobing Sun, Bin Li, Xintong Lu, Hongjing Guo. MULAPI: Improving API method recommendation with API usage location. The Journal of Systems & Software 2018;142:195-205. https://doi.org/10.1016/j.jss.2018.04.060.
6. Reliability of technical systems and technological risk. 2014. URL http://www.obzh.ru/nad/4-3.html.
7. Swift. 2018. URL https://developer.apple.com/swift.
8. Andrea Mario Lufino. What exactly is Xcode? 2017. URL https://www.quora.com/What-is-Xcode.
9. Tools you'll love to use. 2015. URL https://developer.apple.com/xcode/ide.
10. A.S. Walter, S.W. Samuel. System reliability theory. 2004. Hoboken, New Jersey.