Software Relialibility Markovian Model Based on Phase-Type Distribution

Mindaugas Brazenas, Eimutis Valakevicius
Department of Mathematical Modeling
Kaunas University of Technology
Kaunas, Lithuania
e-mail: mindaugas.brazenas@yahoo.com; eimval@ktu.lt

***Abstract*— The paper focuses on creating of a software reliability model based on phase type distribution. Usually, the length of intervals between the moments of fault detection and correction have unknown distributions. In this paper, a new approach how to approximate any distribution of positive random variable by mixture and convolution of exponential phases, known as the general type of phase-type distribution, is proposed. The optimization algorithm of Local Unimodal Sampling (LUS) is applied to estimate parameters of phase-type distribution. After such procedure, the dynamics of a software reliability model can be described by a continuous time absorbing Markov chain. The probabilities of the resulting absorbing Markov chain are used to compute performance measures of the software reliability model.**

Keywords-software reliability model; phase–type distribution; absorbing Markov chain; performance measures.

I. Introduction

Software reliability is the failure probability of the software under investigation. The situation on creating software reliability models is clearly explained in the following citation from the Wikipedia: “Over 225 models have been developed since early 1970s, but how to quantify reliability still remains unsolved. There is no single model which can be used in every situation. There is no model which is either complete or fully developed” [1].

A software reliability model allows forecasting the software reliability at any moment of time. One of the important problems in creating models is an assumption about distribution of the length of intervals between the moments of fault detection. Some of authors assumes that the length of intervals is distributed according to the exponential law [2][3]. For example, the model developed by Moranda and Jelinski [4] assumes an exponential time between failures having parameter that time intervals of detection software faults follow exponential law with the parameter proportional to the number of faults remaining in the system. The similar assumptions are used in [5][6]. Recently**,** non-homogenous Poisson processes became popular for describing stochastic behavior of the number of detected faults, because of their simplicity [7][8][9]. Beside the mentioned distributions, other models that are based on Weibull [10], hyper geometric [11], Pareto [12] and other distributions [13] are investigated.

The use in the software reliability model of any non-exponential distributions is complicated from the computing point of view. Therefore, in this paper, a novel approach to apply a convolution and mixture of exponential distributions, called the Phase-Type (PH) distribution, to approximate time distributions of fault detection and fixing is suggested. It is known that the PH distribution can approximate an arbitrary probability distribution of a positive random variable with an arbitrary accuracy by adjusting the phase structure [14]. Some authors use concrete structure of PH distributions, such as Erlang and hyper exponential [15], Cox [16], or others. The concrete structure of the PH distribution may not approximate the desired distribution with the required accuracy. Many models have been utilised for evaluating the quality of a software using reliability but very little focus on general type of three phase distribution. Hence, this paper mainly focuses on this direction. Using such distribution, the performance of the model can be described by an absorbing Markov chain [14].

The paper is organized as follows. Section II gives description of software reliability model under consideration. Section III describes the algorithm for finding the structure and parameters of approximating PH distribution. The algorithm for constructing the set of all possible states of the system and transition matrix between states is given in Section IV. The modelling results are presented in Section V. The paper is concluded in Section VI.

II. Description of the Model

Let us describe the conceptual model of a software reliability model. Say, that the software contains a fixed number of faults Fc (fault count). Assume that the fault detection time follows some distribution $F^{(a)}(x)$ and fixing time of detected faults obeys another distribution law $F^{(b)}(x)$. The modelling process can be represented as the queuing system (see Figure 1).

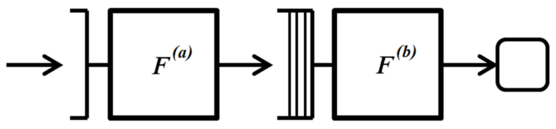

Figure 1. The process of identifying and fixing faults in software.

It is proposed to approximate any general distribution of a positive random variable by the general phase-type distribution (GPH). The phase-type (PH) distribution is defined as the absorbing time distribution of Continuous-Time Markov Chain (CTMC). The detected fault enters the queue if the previously detected one is not fixed yet. The process of detecting and fixing faults ends when all the faults are identified and fixed. The developed software reliability model gives probabilistic measures of the process.

III. Parameter Estimation of The Phase-type Distributions

Parameter estimation of general phase-type distribution is one of the most challenging problems.

The precision of approximation of non-markovian model M by markovian one M^* depends on how well distributions $F^{(a)}$, $F^{(b)}$ are approximated by phase-type distributions $PH^{(a)}$, $PH^{(b)}$. There are several methods to search for optimal phase-type distribution parameters: moment matching method [17][18], expectation maximization method [19][20][21][22], and others. We will search for the optimal parameters by employing a vector optimization algorithm.

The phase-type distribution $PH(\boldsymbol{\pi}, T)$, which has three exponential phases (see Figure 2),

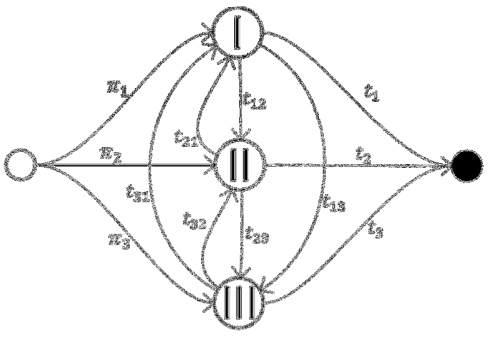

Figure 2. The general structure of PH distribution with three phases.

is determined by 12 variables :

$$\pi = [\pi_1, \pi_2, \pi_3], T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \quad (1)$$

The coordinate $\pi_i, i = 1,2,3$ of the vector $\boldsymbol{\pi}$ (1) denotes the probability of process starting in ith phase. The intensity rates of transition from one phase to other are defined in matrix T. For example, the value t_{13} indicates the average transition number from the first phase to the third one per unit of time. The auxiliary vector $\boldsymbol{t} = [t_1, t_2, t_3]$: $\boldsymbol{t} = (I - T)\mathbf{1}$ denotes rates the absorbing state (black circle) from each phase is reached. For example, the value t_2 indicates the rate of transition from second phase to absorbing state. The transition rates satisfy the following equalities and unequalities

$$\begin{gathered} t_{11}, t_{22}, t_{33} < 0, \\ t_{12}, t_{13}, t_{21}, t_{23}, t_{31}, t_{32}, t_1, t_2, t_3 \geq 0, \\ t_{11} = -(t_{12} + t_{13} + t_1), \\ t_{22} = -(t_{21} + t_{23} + t_2), \\ t_{33} = -(t_{31} + t_{32} + t_3). \end{gathered} \quad (2)$$

The problem of finding optimal parameters of $PH(\boldsymbol{\pi}, T)$ is transformed to a problem of finding the vector $\boldsymbol{x}^* \in [\boldsymbol{l}, \boldsymbol{u}]$, such that $\forall \boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}] : g(\boldsymbol{x}^*) \leq g(\boldsymbol{x})$. Here $g(\cdot)$ is an objective function; $\boldsymbol{l}, \boldsymbol{u}$ – lower and upper bounds of vector $\boldsymbol{x}$. The mapping of vector $\boldsymbol{x}$ to the set of parameters of the phase-type distribution $PH(\boldsymbol{\pi}, T)$ is carried out in the following way

$$\begin{gathered} \boldsymbol{x} = (x_1, x_2, \dots, x_{12}) \to \boldsymbol{\pi} := \frac{[x_1, x_2, x_3]}{x_1 + x_2 + x_3}, \\ t_1 := x_4, t_{12} := x_5, t_{13} := x_6, \\ t_{21} := x_7, t_2 := x_8, t_{23} := x_9, \\ t_{31} := x_{10}, t_{32} := x_{11}, t_3 := x_{12}. \end{gathered} \quad (3)$$

The objective function, to be minimized, is defined as an area between the density functions $f(x; \boldsymbol{\theta})$ and $f_{PTD}(x; \boldsymbol{\pi}, T)$, as

$$S = \int_0^\infty |f_{PH}(x; \boldsymbol{\pi}, T) - f(x; \boldsymbol{\theta})| \, dx \quad (4)$$

The estimation of S (4) is obtained by the following expression

$$\hat{S}(x_{end}, \Delta x; \boldsymbol{\pi}, T) = \sum_{k=1}^{n} |f_{PH}(x_k; \boldsymbol{\pi}, T) - f(x_k; \boldsymbol{\theta})| \Delta x,$$

$$x_k = (k - 0.5)\Delta x, \; n = \left\lfloor \frac{x_{end}}{\Delta x} \right\rfloor \quad (5)$$

where: x_{end} denotes the end value of discretization and Δx – the step of discretization. After the discretization the objective function (4) has the form

$$g(\boldsymbol{x}) := \hat{S}(x_{end}, \Delta x; \boldsymbol{\pi}, T); \; \boldsymbol{x} \to \boldsymbol{\pi}, T. \quad (6)$$

The lower and upper bounds of $\boldsymbol{x}$ are defined as

$$\begin{gathered} \boldsymbol{l} = \Big[\underbrace{0, 0, \dots, 0}_{12} \Big], \\ \boldsymbol{u} = \Big[1, 1, 1, \underbrace{\lambda_{max}, \lambda_{max}, \dots, \lambda_{max}}_{9} \Big], \end{gathered} \quad (7)$$

where λ_{max} – is the maximal transition rate. Using the mapping $\boldsymbol{x} \to \boldsymbol{\pi}, T$, the optimal parameters of the phase-type distribution are obtained from solution $\boldsymbol{x}^*$, which is given by a certain optimization algorithm.

IV. SYSTEM STATE GRAPH CONSTRUCTION ALGORITHM

The scheme of the process of detecting and fixing faults in software after approximating the arbitrary distributions by PH distributions is represented in Figure 3.

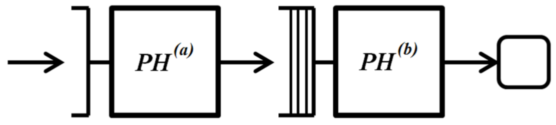

Figure 3.The process of detecting and fixing faults in software after distribution approximation.

The algorithm for constructing the set of states and the transition matrix for markovian model M^* **is** described in this section. The set of states of the system is defined by the vector $\boldsymbol{v} = (k_1, k_2, k_3, k_4)$, where $k_1 \in \{1,2,3\}$ denotes the index of an active phase of $PH^{(a)}$ and $k_1 = 0$ indicates that there is any active phase in $PH^{(a)}$; $k_2 \in \{0,1,\dots,Fc-1\}$ denotes a number of detected faults which are waiting in the queue; $k_3 \in \{1,2,3\}$ denotes the index of an active phase of $PH^{(b)}$, $k_3 = 0$ indicates that there is any active phase in $PH^{(b)}$;$k_4 \in \{0,1,\dots,Fc\}$ – denotes a number of fixed faults. The number of detected faults in the state $\boldsymbol{v}$ is defined according the formula

$$c(\boldsymbol{v}) = \mathbf{1}_{\{k_1>0\}} + k_2 + \mathbf{1}_{\{k_3>0\}} + k_4. \qquad (8)$$

All the states are enumerated by the mapping

$$I_v = \omega(\boldsymbol{v}) \in \{0,1,\dots,r_{max}-1\}, \qquad (9)$$

where r_{max} denotes the number of states: $r_{max} = 16Fc(Fc+1)$. The mapping $\omega(\boldsymbol{v})$ is defined as

$$\omega(\boldsymbol{v}) = (Fc+1)(4k_1Fc + 4k_2 + k_3) + k_4. \qquad (10)$$

The inverse mapping $\omega^{-1}(I_v)$ is obtained by the formulas

$$\omega^{-1}(I_v) = \big(c, b\%Fc, a\%4, I_v\%(Fc+1)\big), \qquad (11)$$

where: $a = \frac{I_v - I_v\%(Fc+1)}{Fc+1}, b = \frac{a - a\%4}{4}, c = \frac{b - b\%Fc}{Fc}$, and % is a reminder operator.

The vector $\boldsymbol{u} = (u_0, u_1, \dots, u_{r_{max}-1})$ of boolean variables (false or true) are used to determine all the possible states of the system. The following sets of states are used: V^{init} contain**s** initial states of the system; V contain**s** all possible states of the system; V^{np} include**s** states which are going to be investigated in the next iteration and V^{tmp} is the temporary set of states obtained from the investigated states after one iteration. Each state contained in these sets is represented by its index I_v.

The transitions rates between all states are stored in the matrix $Q^+ = \{q_{ij}^+\}$; $i,j = \overline{0, r_{max}-1}$.

The algorithm for generating the set of possible states of the system consists of the following steps:

1) *Mark all the states as not investigated:*

$$u_{I_v} := false \;(\,the\ state\ is\ not\ investigated\ yet), \quad I_v = \overline{0, r_{max}-1}$$

2) *Calculate the initial probability vector :*

$$\pi_{I_v}^+ := 0, I_v = \overline{0, r_{max}-1}$$
$$\pi_{\omega((i,0,0,0))}^+ := \pi_i^{(a)}, i = 1,2,3$$

3) *Determine the initial states of the system:*

$$i = 1,2,3{:}\ \pi_i^{(a)} > 0 \;\Rightarrow V^{init} := V^{init} \cup \omega\big((i,0,0,0)\big)$$

4) *Determine the initial values of the sets:*

$$V := V^{init}, V^{tmp} := V^{init}$$
$$\{q_{ij}^+\} := 0;\ i,j = \overline{0, r_{max}-1}$$

5) *Determine the states that have to be investigated:*

$$V^{np} := V^{tmp}$$

Clear the temporary set $V^{tmp} := \emptyset$

Let $\{V^{np}\}_p$ be an p-th element of the set V^{np}.

$$p := 1$$

6) *Find the coordinates of the state vector to be investigated:*

$$\boldsymbol{v} := \omega^{-1}\big(\{V^{np}\}_p\big), I_v := \omega(\boldsymbol{v})$$

7) *Let us denote the set of the state vectors*, that can be reached from the state $\boldsymbol{v}$**,** by $V^* := \emptyset$ and the set of transition rates, at which all states in set V^* are reached from the state $\boldsymbol{v}$**,** by $\lambda^* = \emptyset$.

8) *Find the elements of the sets* $V^* = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_k\}$ and $\lambda^* = \{\lambda_1, \lambda_2, \dots, \lambda_k\}$. The algorithm will be described later.

9) *Update the transition rates matrix:*

$$q_{I_v I_{v_i}}^+ := \lambda_i;\ I_{v_i} = \omega(\boldsymbol{v}_i), i = \overline{1,k}$$

10) *Include not yet investigated states into the sets* $V$and V^{tmp} (see Figure 4)

```
for i from 1 to k
    $I_{v_i} := \omega(\boldsymbol{v}_i)$
    if $u_{I_{v_i}} = false$ then
      $V^{tmp} := V^{tmp} \cup \{I_{v_i}\}, V := V \cup \{I_{v_i}\}$
    endif
  endfor
```

Figure 4. Pseudocode for including not investigated states into the sets $V$and V^{tmp}.

11) *Mark the current state $\boldsymbol{v}$ as investigated:*

$$u_{I_v} := true$$

12) *If* $p < card(V^{np})$ then $p := p + 1$ and go to step 6.

13) *If* $V^{tmp} \neq \emptyset$ go to step 5.

14) *Create the final transition rates matrix* $Q = \{q_{ij}\}$ and the initial probability vector $\boldsymbol{\pi}(0)$ from $Q^+$and$\boldsymbol{\pi}^+(0)$ (see Figure 5).

```
for i from 0 to $card(V) - 1$
  $I := \omega(\{V\}_i)$,
  $\pi_i := \pi_I^+$
  for j from  0 $card(V) - 1$
    $J := \omega(\{V\}_j)$,
    $q_{ij} := q_{IJ}^+$
  endfor
endfor
```

Figure 5. Pseudocode for creating final transition rates matrix and initial probability vector.

15) *The end of the algorithm.*

The explanation of the step 8 in detail follows. Denote by the vector $\boldsymbol{v} = (k_1, k_2, k_3, k_4)$ the state of the system. There are four events that make the system to change the state: e$_1$ – the change of an active phase in $PH^{(a)}$, e$_2$ – the detection of fault, e$_3$ – the change of an active phase in $PH^{(b)}$ and e$_4$ – the correction of fault. The pseudocode needed to process these events is shown in the Figures 6, 7, 8, and 9.

```
for i from 1 to 3
  if $k_1 > 0, t_{k_1 i}^{(a)} > 0$ then
    $V^* := V^* \cup \{(i, k_2, k_3, k_4)\}$,
    $\lambda^* := \lambda^* \cup \{t_{k_1 i}^{(a)}\}$
  endif
endfor
```

Figure 6. Pseudocode for processing the e$_1$ event.

```
if $k_3 = 0$ then
  if $c(\boldsymbol{v}) < Fc$ then
    for j from 1 to 3
      for s from 1 to 3
        if $k_1 > 0, t_{k_1}^{(a)} > 0, \pi_j^{(b)} > 0, \pi_s^{(a)} > 0$ then
          $V^* := V^* \cup \{(s, 0, j, k_4)\}$,
          $\lambda^* := \lambda^* \cup \{t_{k_1}^{(a)} \pi_j^{(b)} \pi_s^{(a)}\}$
        endif
      endfor
    endfor
  else
    for j from 1 to 3
      if $k_1 > 0, t_{k_1}^{(a)} > 0, \pi_j^{(b)} > 0$ then
        $V^* := V^* \cup \{(0, 0, j, k_4)\}$,
        $\lambda^* := \lambda^* \cup \{t_{k_1}^{(a)} \pi_j^{(b)}\}$
      endif
    endfor
  endif
elseif $k_3 > 0$ then
  if $c(\boldsymbol{v}) < Ec$ then
    for s from 1 to 3
      if $k_1 > 0, t_{k_1}^{(a)} > 0, \pi_s^{(a)} > 0$  then
        $V^* := V^* \cup \{(s, k_2 + 1, k_3, k_4)\}$,
        $\lambda^* := \lambda^* \cup \{t_{k_1}^{(a)} \pi_s^{(a)}\}$
      endif
    endfor
  else
    if $k_1 > 0, t_{k_1}^{(a)} > 0$ then
      $V^* := V^* \cup \{(0, k_2 + 1, k_3, k_4)\}$,
      $\lambda^* := \lambda^* \cup \{t_{k_1}^{(a)}\}$
    endif
  endif
endif
```

Figure 7. Pseudocode for processing the e$_2$ event.

```
for i from 1 to 3
  if $k_3 > 0, t_{k_3 i}^{(b)} > 0$ then
    $V^* := V^* \cup \{(k_1, k_2, i, k_4)\}$,
    $\lambda^* := \lambda^* \cup \{t_{k_3 i}^{(b)}\}$
  endif
endfor
```

Figure 8. Pseudocode for processing the e$_3$ event.

```
if $k_2 = 0$ then
  if $k_3 > 0, t^{(b)}_{k_3} > 0$ then
    $V^* := V^* \cup \{(k_1, 0,0, k_4 + 1)\}$,
    $\lambda^* := \lambda^* \cup \left\{t^{(b)}_{k_3}\right\}$
  endif
else
  for s from 1 to 3
    if $k_3 > 0, t^{(b)}_{k_3} > 0, \pi^{(b)}_s > 0$ then
      $V^* := V^* \cup \{(k_1, k_2 - 1, s, k_4 + 1)\}$,
      $\lambda^* := \lambda^* \cup \left\{t^{(b)}_{k_3} \pi^{(b)}_s\right\}$
    endif
  endfor
endif
```

Figure 9. Pseudocode for processing the e$_4$ event

V. MODELING RESULTS

Assume that a software program contains 10 faults (*Fc*=10) and suppose that the length of intervals between the moments of fault detection has the following Weibull density function

$$f^{(a)}(x) = \frac{4}{3}\left(\frac{4}{3}\right)^{-0.2} e^{-\frac{x}{0.6}},\ x \geq 0. \qquad (12)$$

The distribution of the length of intervals between the moments of fixing faults has the following Weibull density function

$$f^{(b)}(x) = \frac{1.3}{1.5}\left(\frac{1.3}{1.5}\right)^{0.3} e^{-\frac{x}{1.3}},\ x \geq 0. \qquad (13)$$

The discretization parameters **are** $x_{end} = 8$, $\Delta x = 0.0625$.

The distributions $f^{(a)}, f^{(b)}$ are approximated by the phase-type distributions $f^{(a)}_{PH}\left(x;\ \boldsymbol{\pi}^{(a)}, T^{(a)}\right)$, $f^{(b)}_{PH}\left(x;\ \boldsymbol{\pi}^{(b)}, T^{(b)}\right)$ with three phases. The following optimal parameters for $f_{PH^{(a)}}\left(x;\ \boldsymbol{\pi}^{(a)}, T^{(a)}\right)$ and $f^{(b)}_{PH}\left(x;\ \boldsymbol{\pi}^{(b)}, T^{(b)}\right)$ density functions are estimated using the optimization algorithm LUS [23] and parameter mapping given in Section III.

$$\hat{S}^{(a)} = 0.012693, \boldsymbol{\pi}^{(a)} \approx [3.456 \cdot 10^{-4}, 0.767, 0.233],$$

$$T^{(a)} \approx \begin{bmatrix} -1.516 & 0.002 & 1.509 \\ 0.584 & -2.348 & 0.005 \\ 1.782 & 7.819 & -15.604 \end{bmatrix},$$

$$\hat{S}^{(b)} = 0.010634, \boldsymbol{\pi}^{(b)} \approx [0.008, 1.397 \cdot 10^{-4}, 0.992],$$

$$T^{(b)} \approx \begin{bmatrix} -1.236 & 0 & 0 \\ 1.913 & -16.338 & 6.770 \\ 0.897 & 0.331 & -1.402 \end{bmatrix}$$

The comparision of means and standard deviations between original and approximated distributions are given in the Tables 1 and 2.

TABLE I. MEAN AND STANDARD DEVIATION OF ORIGINAL, DISCRETIZED AND APPOXIMATING DISTRIBUTIONS OF TIME FOR FAULT DETECTION

	Distri-bution $F^{(a)}$	**Discretized distribution $\hat{F}^{(a)}$(deviation,%)**	**Phase-type distribution $PH^{(a)}$(deviation,%)**
Mean	0.6798	0.6769 (0.43%)	0.6958 (2.35%)
Standard deviation	0.8569	0.8364 (2.39%)	0.8588 (0.22%)

TABLE II. MEAN AND STANDARD DEVIATION OF ORIGINAL, DISCRETIZED AND APPOXIMATING DISTRIBUTIONS OF TIME FOR FAULT CORRECTION

	Distri-bution $F^{(b)}$	**Discretized distribution $\hat{F}^{(b)}$(deviation,%)**	**Phase-type distribution $PH^{(b)}$(deviation,%)**
Mean	1.3854	1.3841 (0.09%)	1.4000 (1.05%)
Standard deviation	1.0747	1.0719 (0.26%)	1.1126 (3.53%)

The graphs of the original and approximated density functions are represented in Figures 10 and 11.

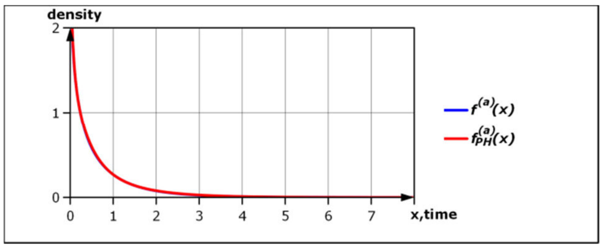

Figure 10. The density functions $f^{(a)}$ (blue) and $f^{(a)}_{PH}$ (red).

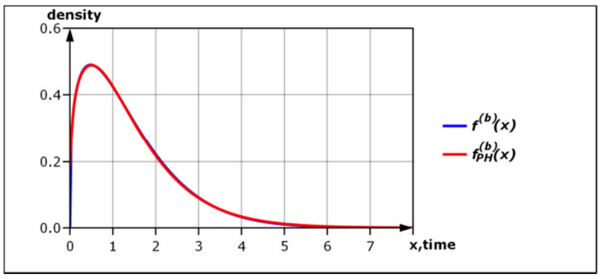

Figure 11. The density functions $f^{(b)}$ (blue) and $f^{(b)}_{PH}$ (red).

The markovian software reliability model has 466 states. The state probabilities are computed by the following formula

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0)e^{Qt}. \quad (14)$$

All possible states are grouped according the number of detected faults that are placed in the queue. The values $L_n(t)$, that there is a certain number n of faults waiting in the queue, are obtained by probability summation within each state group

$$L_n(t) = \sum_{i,k_2=n} \pi_i(t),$$
$$\omega^{-1}(\{V\}_i) = (k_1, k_2, k_3, k_4), n = \overline{0, Fc-1}, \quad (15)$$

where $\{V\}_i$ is the i^{th} element in the set V. The graph of the values $L_n(t)$ is shown in Figure 12.

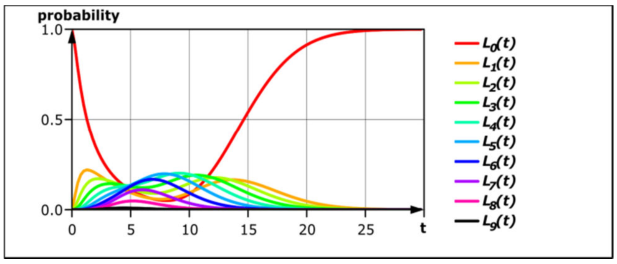

Figure 12. The probability functions (of time) of certain number of faults waiting in the queue.

Similarly, all possible states of the system are grouped according the number of fixed faults. The values $E_n(t)$, that there is a certain number n of fixed faults, are obtained by probability summation within each state group

$$E_n(t) = \sum_{i,k_4=n} \pi_i(t),$$
$$\omega^{-1}(\{V\}_i) = (k_1, k_2, k_3, k_4), n = \overline{0, Fc}, \quad (16)$$

where $\{V\}_i$ is the i^{th} element in the set V.
The graph of values $E_n(t)$ is shown in Figure 13.

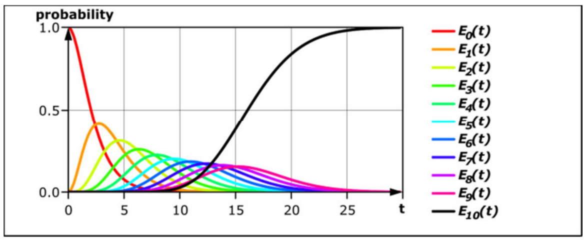

Figure 13. The probabilitiy functions (of time) of a certain number of fixed faults.

The density function of time of fixing all errors is shown in Figure 14.

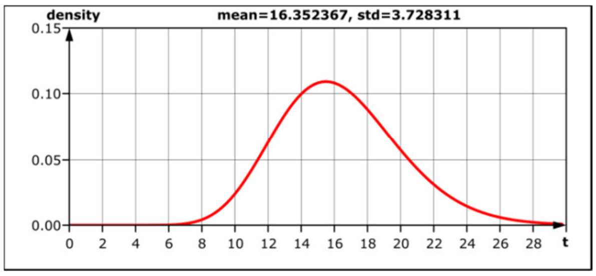

Figure 14. Density function of distribution of time necessary to fix all 10 faults.

The most probable that the time needed to fix all 10 faults is about 16 units of time.

VI. CONCLUSION AND FUTURE WORKS

In this article, a continuous time absorbing Markov chain model of software reliability was proposed. Non-markovian distributions of the length of intervals between the moments of fault detection and correction are approximated by the general phase-type distributions with three phases. The model generalizes other software reliability models in which various types of distributions are used. The probabilistic measures of detecting and fixing faults of created software are presented. The proposed model can be useful in estimating and monitoring software reliability, which is viewed as a measure of software quality. Therefore, it can be concluded that this model is more realistic then others for a detection of software faults.

In the future, the following modified software reliability model will be created and investigated. The detected fault must be fixed before searching for the next one with the assumption that the distributions can change depending on number of detected/fixed faults. Examples of the application how a model help to have better software will be added.

REFERENCES

[1] List, "List of software reliability models", available: http://en.wikipedia.org/wiki/List_of_software_reliability _models [retrieved: July, 2014].

[2] B. Zachariah and R. N. Rattihalli, "Failure size proportional models and an analysis of failure detection abilities of software testing strategies," IEEE Transactions on Reliability, vol. 56, n. 2, 2007, pp. 246-253.

[3] Y. P. Wu, Q. P. Hu, M. Xie, and S. H. Ng, "Modeling and analysis of software fault detection and correction process by considering time dependency," IEEE Transactions on Reliability, vol. 56, n. 4, 2007, pp. 629-642.

[4] P. Moranda and Z. Jelinski, "Final report on software reliability study, MADC report number 63921," McDonnell Douglas Astronautics Company, 1972.

[5] J. Musa, A. Iannino, and K. Okumoto, "Software reliability measurement, prediction, application," McGraw Hill, NewYork, 1987.

[6] A. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance

measures," IEEE Transactions on Reliability, R–28 (3), 1979, pp. 206–211.

[7] A. L. Goel, "Software reliability models: assumptions, limitations and applicability," IEEE Trans. Software Eng., SE-11, 1985, pp. 1411–1423.

[8] S. S. Gokhale and K. S. Trivedi, "Log-logistic software reliability growth model," Proc. 3rd IEEE Int'l. High-Assurance Systems Eng., Symp IEEE CS Press, 1998, pp. 34–41.

[9] M. Ohba, "Inflection S-shaped software reliability growth model," Stochastic Models in Reliability Theory, (S. Osaki and Y. Hatoyama, eds.), Springer-Varlag, Berlin, Germany, 1984, pp. 144–165.

[10] S. Quadri and N. Ahmad, "Software reliability growth modeling with new modified Weibul testing-effort and optimal release policy," International Journal of Computer Applications, Vol. 6, No.2, 2010, pp. 1-10.

[11] Y. Tohma, R. Jacoby, Y. Murata, and M. Yamamoto, "Hypergeometric distribution model to estimate the number of residual software faults," Proceedings of the 13th Annual International Computer Software and Applications, Conference (COMPSAC'89), 1989, pp. 610-617.

[12] Y. Aamsidhar, Y. Srinivas, and A. Brahmini, "Software reliability growth model based on Pareto type III distribution," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 2, No.6, 2013, pp. 2694-2698.

[13] S. Inoue and S. Yamada, "Lognormal process software reliability modeling with testing-effort," Journal of Software Engineering and Application, Vol. 6, 2013, pp. 8-14.

[14] M. F. Neuts, "Matrix-Geometric Solutions in Stochastic Models: an Algorithmic Approach", Dover Publications Inc., 1981.

[15] H. Okamara and T. Dohi, "Building Phase-Type software reliability models', 17th International Symposium on Software Reliability Engineering (ISSRE'06), November 2007, pp. 289-298.

[16] V. Bubnov, A. Tyrva, and A. Khomonenko, "Model of reliability of the software with Coxian distribution of length of intervals between the moments of detection errors," IEEE 35th Annual Computer Software and Applications Conference Workshops, COMPSACW, 2011, pp. 310-314.

[17] A. Bobbio, A. Horváth, and M. Telek, "Matching three moments with minimal acyclic phase type distributions," Stochastic Models, 21, 2005, pp. 303-326.

[18] H. András and T. Miklós, "Matching More Than Three Moments with Acyclic Phase Type Distributions," Stochastic Models, 23, 2007, pp. 167-194.

[19] S. Asmussen, O. Nerman, and M. Olsson, "Fitting Phase-Type Distributions via the EM Algorithm,“ Scandinavian Journal of Statistics, Vol. 23, No. 4, December 1996, pp. 419-441.

[20] R. Sadre and B. R. Haverkort, "Fitting Heavy-Tailed HTTP Traces with the New Sratified EM-Algoritm," IT-NEWS 2008 – 4th International Telecommunication Networking Workshop on QoS Multiservice IP Networks, 2008, pp. 256-261.

[21] A. Risha, V. Diev, and E. Smirni, "An EM-based technique for approximating long-tailed data sets with PH distributions," Performance Evaluation, 55 (2), 2004, pp. 147–164.

[22] L. J. R. Esparza, "Maximum likelihood estimation of pahse-type distributions," Kongens Lyngby 2010, IMM-PHD-2010-245.

[23] E. Pedersen and A. J. Chipperfield, "Local Unimodal Sampling", Hvass Laboratories Technical Report no. HL0801, 2008.