UDC 004.93

ANDROID SOFTWARE AGING AND REJUVENATION MODEL CONSIDERING THE BATTERY CHARGE

Yakovyna V. S. – Dr. Sc., Full Professor of Artificial Intelligence Department, Lviv Polytechnic National University, Lviv, Ukraine; Faculty of Mathematics and Computer Science, University of Warmia and Mazury in Olsztyn, Poland.

Uhrynovskyi B. V. – Assistant of Software Department, Lviv Polytechnic National University, Lviv, Ukraine.

ABSTRACT

Context. A feature of mobile systems is their dependence on battery charge, which is an important factor when planning various processes, in particular when planning time of performing software rejuvenation procedure.

Objective. The goal of this article is to develop a model of software aging process with performing rejuvenation procedure for the Android operating system considering the factor of battery charge.

Method. A complex model based on Continuous-Time Markov Chains is proposed, which combines the software aging and rejuvenation model, the user behavior model and consider battery charge factor. A graph of states and transitions describing a complex model is constructed. Based on the formed graph the system of differential equations is written. The system was calculated using the 4th order Runge-Kutta method. The optimal time for the rejuvenation procedure can be determined when rejuvenation will not interfere with the user and will be performed before the battery is fully discharged, ie when the probability of the system being in these states is the lowest.

Results. The simulation of the developed model for test values of transition rates is performed. Considering the battery charge model allows to avoid planning the rejuvenation procedure at a time when the mobile device is likely to have a low charge or be completely discharged.

Conclusions. The proposed model based on the Markov chain allows to predict the start time of software rejuvenation procedure, considering both user behavior and battery level, which can have a significant impact on the predicted time. Also, the early implementation of the rejuvenation procedure may have the effect of reducing the system workload and delaying the discharge of the device, which should be checked in further studies. The expediency and importance of the consideration of battery charge factor and the need for further study of the proposed software aging and rejuvenation model are substantiated.

KEYWORDS: software aging, software rejuvenation, Markov Chains, Android.

ABBREVIATIONS

OS is an operating system;

UI is a user interface;

AY is an “Active Young” state;

RA is a “Rebirth Active” state;

ARe is an “Active Recovering” state;

AYS(L)p is an “Active Young Stable (Low) Power” state;

AOS(L)p is an “Active Old Stable (Low) Power” state;

AReS(L)p is an “Active Recovering Stable (Low) Power” state;

ARS(L)p is an “Active Rebirth Stable (Low) Power” state;

SYS(L)p is a “Sleep Young Stable (Low) Power” state;

SOS(L)p is a “Sleep Old Stable (Low) Power” state;

SReS(L)p is a “Sleep Recovering Stable (Low) Power” state;

SRS(L)p is a “Sleep Rebirth Stable (Low) Power” state;

Op is an “Off Power” state.

NOMENCLATURE

λ is a transition rate between states;

λ_{AS} is a transition rate from “Active” state to “Sleep” state;

λ_{SA} is a transition rate from “Sleep” state to “Active” state;

λ_{YO} is a transition rate from “Young” state to “Old” state;

λ_{ORe} is a transition rate from “Old” state to “Recovering” state;

λ_{ReY} is a transition rate from “Recovering” state to “Young” state;

λ_{YR} is a transition rate from “Young” state to “Rebirth” state;

λ_{RY} is a transition rate from “Rebirth” state to “Young” state;

λ_{HpLp} is a transition rate from “High Power” state to “Low Power” state;

λ_{LpCh} is a transition rate from “Low Power” state to “Charging” state;

λ_{ChHp} is a transition rate from “Charging” state to “High Power” state;

λ_{LpOp} is a transition rate from “Low Power” state to “Off Power” state;

λ_{SpLp} is a transition rate from “Stable Power” state to “Low Power” state;

λ_{LpSp} is a transition rate from “Low Power” state to “Stable Power” state;

t is a time;

P(t) is the probabilities vector of the system in different states at time *t*;

$P_{AYS(L)p}(t)$ is the probability of a system being at time t in the “Active Young Stable (Low) Power” state;

$P_{AOS(L)p}(t)$ is the probability of a system being at time t in the “Active Old Stable (Low) Power” state;

DOI 10.15588/1607-3274-2021-4-13

$P_{AReS(L)p}(t)$ is the probability of a system being at time t in the “Active Recovering Stable (Low) Power” state;

$P_{ARS(L)p}(t)$ is the probability of a system being at time t in the “Active Rebirth Stable (Low) Power” state;

$P_{SYS(L)p}(t)$ is the probability of a system being at time t in the “Sleep Young Stable (Low) Power” state;

$P_{SOS(L)p}(t)$ is the probability of a system being at time t in the “Sleep Old Stable (Low) Power” state;

$P_{SReS(L)p}(t)$ is the probability of a system being at time t in the “Sleep Recovering Stable (Low) Power” state;

$P_{SRS(L)p}(t)$ is the probability of a system being at time t in the “Sleep Rebirth Stable (Low) Power” state;

$P_{Op}(t)$ is the probability of a system being at time t in the “Off Power” state;

T_{avg} is an average transition time.

INTRODUCTION

Software aging process is a phenomenon of performance deterioration and failures rate increase caused by the accumulation of software and system errors that are unpredictable and their effects can occur with delays in systems that run for a long time without reboots [1–7]. Aging-related errors are memory leaks, rounding errors, etc. The phenomenon of software aging has been widely studied in various systems and applications, such as Linux, Apache server, middleware [3–6], Android OS [8–10].

Software rejuvenation procedure is a proactive technique to prevent and delay software aging [2]. This approach involves the planned regular or irregular cleaning of the system internal state in such a way as to reduce the number of accumulated errors. Software rejuvenation can be applied to different levels of the system, such as system processes and user applications, data objects within processes, or the entire operating system.

Mobile systems have specific characteristics of operation and use by the user, compared to server or personal computer systems. The previous article [10] describes some features of mobile platforms, as well as the relevance of studying the phenomenon of software aging in mobile systems. Software aging in Android OS manifests as slow response of the user interface after prolonged use of a mobile phone, which is common issue for most users. Also, Android is one of the most popular systems for mobile phones and occupies 84.1% of the market as of 2020 [11]. Due to the urgency of the software aging problem in mobile systems and the popularity of Android OS, an important task is to develop mathematical and algorithmic tools to combat the phenomenon of software aging, in particular modeling of the phenomenon taking into account different factors allows to plan the rejuvenation procedure.

The object of the study is the process of software aging.

The subject of the study is a model of the software aging process for mobile OS with performing rejuvenation procedure.

The purpose of the work is to develop a model of software aging process for Android OS with performing rejuvenation procedure, which takes into account the model of mobile device usage by the users and battery charge level.

© Yakovyna V. S., Uhrynovskyi B. V., 2021
DOI 10.15588/1607-3274-2021-4-13

1 PROBLEM STATEMENT

The rejuvenation procedure is used to counteract the phenomenon of software aging. Performing this procedure requires predicting the occurrence of software aging and planning the time of rejuvenation based on the predicted data. The model of the rejuvenation process should take into account the specific factors and features of the Android OS, as well as data that will objectively reflect the status and dynamics of changes in system performance to be able to make accurate predictions.

Analytical description of software aging process with rejuvenation procedure will be presented in the form of Continuous-Time Markov Chains. Steps needed to build a model and perform calculations to determine the optimal rejuvenation time:

– build and describe the model of software aging process and rejuvenation procedure, the model of the battery charge states and the model of mobile phone user activity;

– identify the states in which the system may be according to considered models and describe the transition rates between these states;

– build a graph of state transitions and describe a system of differential equations based on this graph;

– calculate a system of differential equations using a test set of transition rates between states and analyze the results of calculations.

Calculating the system of differential equations will determine the probabilities of the mobile OS in different states at a particular time. To determine the optimal time to perform the rejuvenation procedure, proposed model allows to find the time when the probability of the system being in a state of active usage and low battery is minimal and aging-related effects are observed.

2 RELATED WORKS

To counteract the phenomenon of software aging in the Android mobile system, various approaches are proposed, both at the level of optimizing the architecture of the developed software [12] and by performing the rejuvenation procedure [13–14].

The article [15] proposes a model, which is a combination of the user behavior model and the software aging model with performing rejuvenation procedure. These models and their combinations are constructed in the form of stochastic Petri nets [16], based on which Continuous-Time Markov Chains were generated [17].

The model of user behavior is shown in Figure 1 and is described by two possible states: “Active” (the user is actively using the phone) and “Sleep” (the phone is in standby mode). Determining whether the device is in a certain state, as well as determining the transition rate between states, is proposed by accumulating statistics on user activity over time.

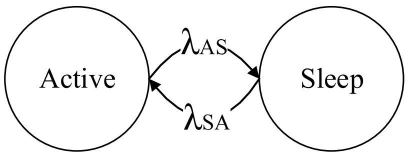


Figure 1 – Mobile device usage model

Software aging model with performing rejuvenation procedure consists of four states: "Young", "Old", "Recovering" and "Rebirth". It is assumed that when the device is turned on, the system is productive and the probability of aging-related failure is very low, i.e., the system is at "Young" state. After prolonged use of the phone, the system goes into "Old" state. The probability of such a transition can be determined by monitoring certain system metrics (such as UI response time and memory usage) and certain thresholds that will be indicators of the transition from "Young" to "Old" state. Switching from "Old" state to "Recovering" state means restarting the mobile device by the user due to critically poor performance or device rebooting due to system aging-related failure. Restart of the device returns the system to "Young" state. The rejuvenation procedure takes place in "Rebirth" state, which can be accessed from "Young" state.

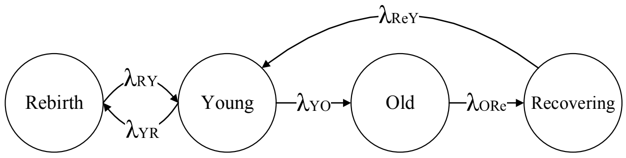


Figure 2 – Software aging model with rejuvenation procedure

The main goal of the model when planning the rejuvenation procedure is to determine the optimal time of transition to "Rebirth" state, which will precede the transition to "Old" state. An important factor in choosing the optimal time is to restrict the performance of the rejuvenation procedure if the user is currently using the phone, i.e., the phone is in "Active" state. This is important limitation because the rejuvenation procedure may have negative impact on user experience or cause a failure when user uses a mobile device, for example, if the rejuvenation procedure involves rebooting system or critical system components.

The model proposed by the authors [15] allowed them to determine the optimal time for the rejuvenation procedure (approximately 30 hours after turning on the phone), although the input data for the model were selected with many simplifications and assumptions. Proposed model can be expanded and improved. This paper proposes new model which considers the battery charge factor, as mobile devices depend on the battery capacity and are vulnerable to increase of intensity of the battery discharge.

3 MATERIALS AND METHODS

The model discussed in the previous section is proposed to be extended by an additional model that describes the battery charge. Two variants of battery charge models can be considered: general and simplified, which will be used in this work. The general model of a battery charge level can be described by means of four states (Fig. 3):

1) High Power – high battery level (for example, battery charge interval 100–25%);

2) Low Power – critically low battery charge at which it is impractical to perform rejuvenation procedure (for example, less than 25%);

3) Charging – the device is charging;

4) Off Power – the device discharged and turned off.

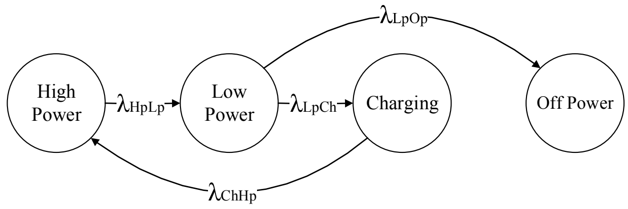


Figure 3 – General model of battery charge

State "Charging" does not actually affect the choice of rejuvenation, as it is impossible to predict the transition to this state or the duration of stay in this state, so the choice to rejuvenate or not still depends on the current battery status. Therefore, it is proposed to simplify this model and replace "High Power" and "Charging" states with the "Stable Power" state, which will mean a sufficient charge of the phone battery for the rejuvenation procedure. It is worth noting that "Charging" state can still be taken into account to adjust and delay the rejuvenation procedure after the initial planning based on the model. The final version of the simplified model is shown in Figure 4.

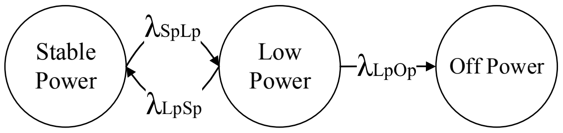


Figure 4 – Simplified model of battery charge

The transitions between states in this model can be calculated based on information about the current state of battery charge and the estimated time to complete discharge. The transition threshold between "Stable" and "Low", "Low" and "Off" states can be expressed as predictions of the discharge or charging time to the appropriate level, for example, 12 hours before the battery is fully discharged, or 3 hours before the battery is fully charged.

The software aging model with rejuvenation, the user behavior model and the model of battery charge were combined into one model, which is shown in Figure 5 and is represented by a Continuous-Time Markov Chains. An important feature of the new model is the impossibility of transition to "Rebirth" state from "Active" and "Low Power" states.

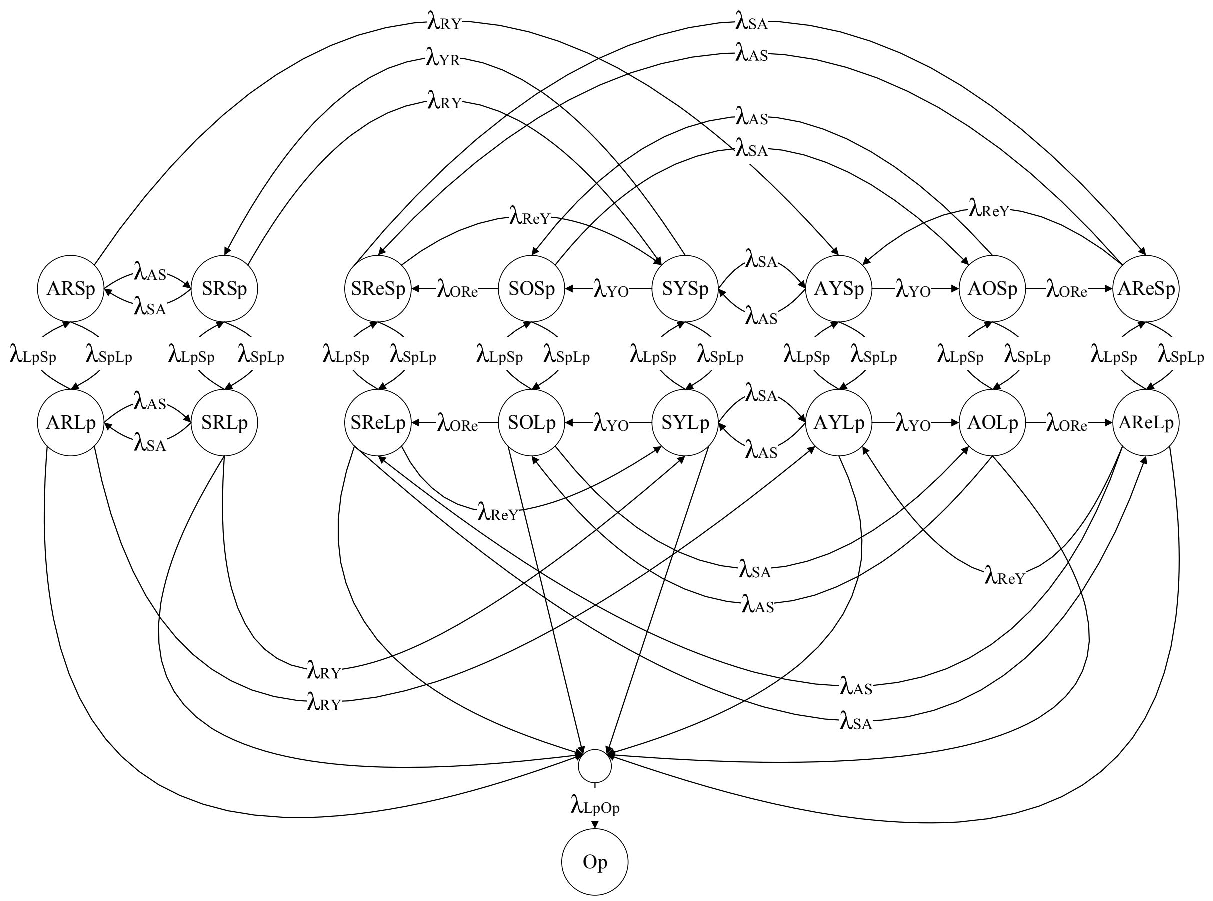


Figure 5 – Continuous-Time Markov Chain for the model of software aging and rejuvenation which takes into account the battery charge

Given the matrix of transition rates between states and the probabilities vector of the system in different states at the initial time, we can calculate the probabilities vector at any subsequent time (1):

$$P(t)=(P_{S\,\mathrm{Re}\,Sp}(t),P_{SOSp}(t),P_{SYSp}(t),P_{SRSp}(t),$$
$$P_{ARSp}(t),P_{AYSp}(t),P_{AOSp}(t),P_{A\,\mathrm{Re}\,Sp}(t),P_{S\,\mathrm{Re}\,Lp}(t),$$
$$P_{SOLp}(t),P_{SYLp}(t),P_{SRLp}(t),P_{ARLp}(t),P_{AYLp}(t), \qquad (1)$$
$$P_{AOLp}(t),P_{A\,\mathrm{Re}\,Lp}(t),P_{Op}(t)).$$

A system of differential equations (2) is written based on the Markov Chain, which can be calculated by numerical methods, in particular by the Runge-Kutta method.

$$\frac{dP_{S\,\mathrm{Re}\,Sp}(t)}{dt}=-(\lambda_{SA}+\lambda_{\mathrm{Re}\,Y}+\lambda_{SpLp})P_{S\,\mathrm{Re}\,Sp}(t)+\lambda_{SA}P_{A\,\mathrm{Re}\,Sp}(t)+\lambda_{O\,\mathrm{Re}}P_{SOSp}(t)+\lambda_{LpSp}P_{S\,\mathrm{Re}\,Lp}(t);$$
$$\frac{dP_{SOSp}(t)}{dt}=-(\lambda_{SA}+\lambda_{O\,\mathrm{Re}}+\lambda_{SpLp})P_{SOSp}(t)+\lambda_{AS}P_{AOSp}(t)+\lambda_{YO}P_{SYSp}(t)+\lambda_{LpSp}P_{SOLp}(t);$$
$$\frac{dP_{SYSp}(t)}{dt}=-(\lambda_{SA}+\lambda_{YO}+\lambda_{YR}+\lambda_{SpLp})P_{SYSp}(t)+\lambda_{AS}P_{AYSp}(t)+\lambda_{S\,\mathrm{Re}\,Sp}P_{\mathrm{Re}\,Y}(t)+\lambda_{SRSp}P_{RY}(t)+\lambda_{LpSp}P_{SYLp}(t); \qquad (2)$$
$$\frac{dP_{SRSp}(t)}{dt}=-(\lambda_{SA}+\lambda_{RY}+\lambda_{SpLp})P_{SRSp}(t)+\lambda_{AS}P_{ARSp}(t)+\lambda_{YR}P_{SYSp}(t)+\lambda_{LpSp}P_{SRLp}(t);$$
$$\frac{dP_{ARSp}(t)}{dt}=-(\lambda_{AS}+\lambda_{RY}+\lambda_{SpLp})P_{ARSp}(t)+\lambda_{SA}P_{SRSp}(t)+\lambda_{LpSp}P_{ARLp}(t);$$

$$\frac{dP_{AYSp}(t)}{dt}=-(\lambda_{AS}+\lambda_{YO}+\lambda_{SpLp})P_{AYSp}(t)+\lambda_{SA}P_{SYSp}(t)+\lambda_{RY}P_{ARSp}(t)+\lambda_{\mathrm{Re}Y}P_{A\mathrm{Re}Sp}(t)+\lambda_{LpSp}P_{AYLp}(t);$$

$$\frac{dP_{AOSp}(t)}{dt}=-(\lambda_{AS}+\lambda_{O\mathrm{Re}}+\lambda_{SpLp})P_{AOSp}(t)+\lambda_{SA}P_{SOSp}(t)+\lambda_{YO}P_{AYSp}(t)+\lambda_{LpSp}P_{AOLp}(t);$$

$$\frac{dP_{A\mathrm{Re}Sp}(t)}{dt}=-(\lambda_{AS}+\lambda_{\mathrm{Re}Y}+\lambda_{SpLp})P_{A\mathrm{Re}Sp}(t)+\lambda_{SA}P_{S\mathrm{Re}Sp}(t)+\lambda_{O\mathrm{Re}}P_{AOSp}(t)+\lambda_{LpSp}P_{A\mathrm{Re}Lp}(t);$$

$$\frac{dP_{S\mathrm{Re}Lp}(t)}{dt}=-(\lambda_{SA}+\lambda_{\mathrm{Re}Y}+\lambda_{LpSp}+\lambda_{LpOp})P_{S\mathrm{Re}Lp}(t)+\lambda_{AS}P_{A\mathrm{Re}Lp}(t)+\lambda_{O\mathrm{Re}}P_{SOLp}(t)+\lambda_{SpLp}P_{S\mathrm{Re}Sp}(t);$$

$$\frac{dP_{SOLp}(t)}{dt}=-(\lambda_{SA}+\lambda_{O\mathrm{Re}}+\lambda_{LpSp}+\lambda_{LpOp})P_{SOLp}(t)+\lambda_{AS}P_{AOLp}(t)+\lambda_{YO}P_{SYLp}(t)+\lambda_{SpLp}P_{SOSp}(t);$$

$$\frac{dP_{SYLp}(t)}{dt}=-(\lambda_{SA}+\lambda_{YO}+\lambda_{LpSp}+\lambda_{LpOp})P_{SYLp}(t)+\lambda_{AS}P_{AYLp}(t)+\lambda_{S\mathrm{Re}Lp}P_{\mathrm{Re}Y}(t)+\lambda_{SRLp}P_{RY}(t)+\lambda_{SpLp}P_{SYSp}(t);$$

$$\frac{dP_{SRLp}(t)}{dt}=-(\lambda_{SA}+\lambda_{RY}+\lambda_{LpSp}+\lambda_{LpOp})P_{SRLp}(t)+\lambda_{AS}P_{ARLp}(t)+\lambda_{SpLp}P_{SRSp}(t);$$

$$\frac{dP_{ARLp}(t)}{dt}=-(\lambda_{AS}+\lambda_{RY}+\lambda_{LpSp}+\lambda_{LpOp})P_{ARLp}(t)+\lambda_{SA}P_{SRLp}(t)+\lambda_{SpLp}P_{ARSp}(t);$$

$$\frac{dP_{AYLp}(t)}{dt}=-(\lambda_{AS}+\lambda_{YO}+\lambda_{LpSp}+\lambda_{LpOp})P_{AYLp}(t)+\lambda_{SA}P_{SYLp}(t)+\lambda_{RY}P_{ARLp}(t)+\lambda_{\mathrm{Re}Y}P_{A\mathrm{Re}Lp}(t)+\lambda_{SpLp}P_{AYSp}(t)$$

$$\frac{dP_{AOLp}(t)}{dt}=-(\lambda_{AS}+\lambda_{O\mathrm{Re}}+\lambda_{LpSp}+\lambda_{LpOp})P_{AOLp}(t)+\lambda_{SA}P_{SOLp}(t)+\lambda_{YO}P_{AYLp}(t)+\lambda_{SpLp}P_{AOSp}(t);$$

$$\frac{dP_{A\mathrm{Re}Lp}(t)}{dt}=-(\lambda_{AS}+\lambda_{\mathrm{Re}Y}+\lambda_{LpSp}+\lambda_{LpOp})P_{A\mathrm{Re}Lp}(t)+\lambda_{SA}P_{S\mathrm{Re}Lp}(t)+\lambda_{O\mathrm{Re}}P_{AOLp}(t)+\lambda_{SpLp}P_{A\mathrm{Re}Sp}(t);$$

$$\frac{dP_{Op}(t)}{dt}=\lambda_{LpOp}(P_{SYLp}(t)+P_{AYLp}(t)+P_{SOLp}(t)+P_{AOLp}(t)+P_{RALp}(t)+P_{RSLp}(t)+P_{A\mathrm{Re}Lp}(t)+P_{S\mathrm{Re}Lp}(t)).$$

The main goal of the model considered in this paper is to predict the optimal time to perform the rejuvenation procedure. Since the time distribution in models based on the Continuous-Time Markov Chain is exponential, the average transition time to a certain state is determined by the formula:

$$T_{avg}=\frac{1}{\lambda}. \qquad (3)$$

By calculating a system of differential equations (2) of model for different λ_{YR} values, the optimal rejuvenation time can be determined when the probability of being in the Active" and "Low Power" states is the lowest, i.e., when the rejuvenation procedure will not interfere with the user and will be performed until the battery is fully discharged.

4 EXPERIMENTS

To verify and analyze the proposed model of software aging with rejuvenation, the system of differential equations (2) was calculated by the 4th order Runge-Kutta method. The calculation of the system of equations for the original model [15] is also performed and it allows us to compare the values of 1/ λ_{YR} of both models and determine the influence of the factor on forecasting the rejuvenation procedure. Transition rates between all states of the system are approximate values which allow to simulate aging process and perform experimental calculations.

In work [15] two periods of user activity are considered, namely day (from 9 a.m. till 10 p.m.) and night (from 10 p.m. till 9 a.m.). Rates of transitions between "Active" and "Sleep" states for these intervals can be determined based on the collected user activity data. For example, you can divide the day into 15-minute time intervals, which will be denoted as "Active" or "Sleep" depending on the average number of events of the device sensors collected over many previous days [15]. In this study, 1 minute is used as a unit of time. The period of user activity in the day for analysis has no fundamental differences, so this paper assumes the verification of the model for the day period. Thus, we assume that the transition rates of λ_{AS} and λ_{SA} are 0.05 (average time of phone usage before entering sleep state is 20 minutes) and 0.02 (average time in sleep state is 50 minutes), respectively.

Transition rates from "Young" to "Old" state and from "Old" to "Recovering" state in [15] are considered equal and are based on increasing the average delay time of the user interface. These transition rates in a real system can be calculated based on the observed aging-related metrics, which show a performance degradation or an increase

DOI 10.15588/1607-3274-2021-4-13

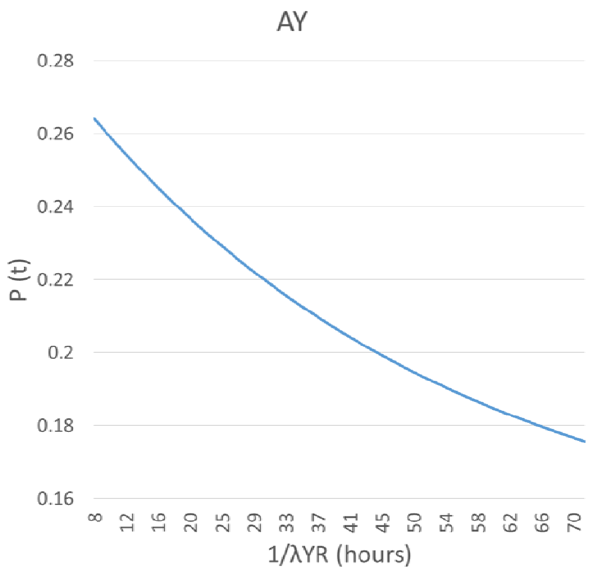
usage of system resources. However, in this paper, it is assumed that the transitions from “Young” to “Old” and from “Old” to “Recovering” last 100 hours, so $\lambda_{YO} = \lambda_{ORe} = 0.00017$. Rejuvenation procedure and recovery of the system after aging can be considered as a process of device rebooting, which lasts about one minute, so $\lambda_{ReY} = \lambda_{RY} = 1$.

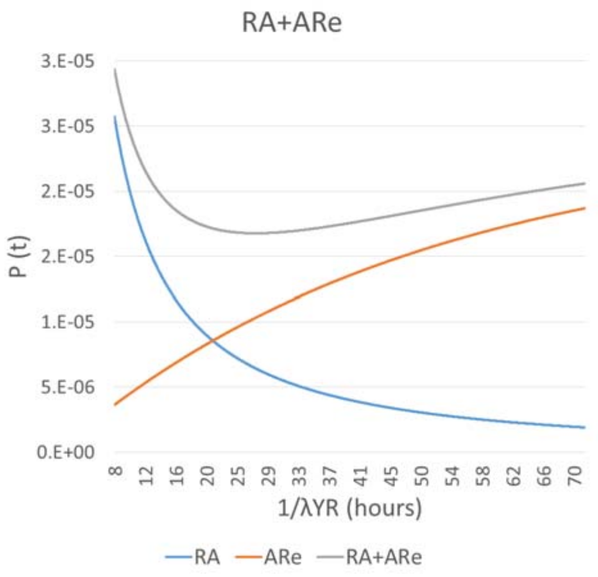
Transitions rates of battery charge model can be obtained based on remaining charge predictions at the time of calculations on the mobile device. In this work, it is assumed that the discharge time of the battery to a low level, and then to complete shutdown is 24 and 1 hour, respectively, i.e., $\lambda_{SpLp} = 0.0007$, $\lambda_{LpOp} = 0.017$. The time required to charge the battery to full level is assumed to be equal to 2 hours, i.e., $\lambda_{LpSp} = 0.0083$.

5 RESULTS

Calculations results of differential equations system of the original model without considering the battery charge are shown in Figure 6. Fig. 6 (a) shows the probability of being in the state AY, and Fig. 6 (b) shows the sum of the probabilities of being in the RA and ARe states.

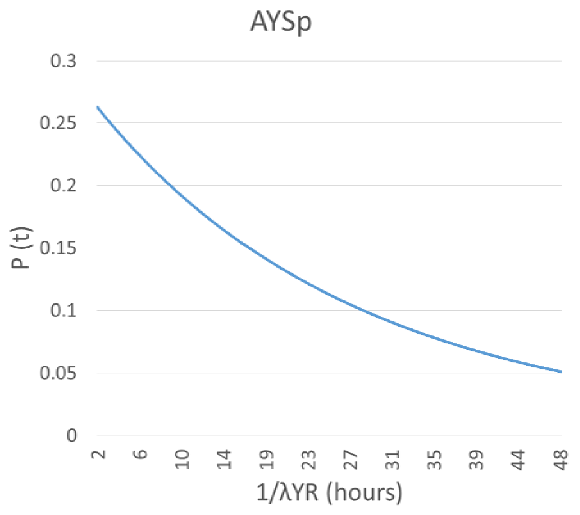
Figure 7 shows the results of calculations of model considering the battery charge. Fig. 7 a shows the probability of being in the state of AYSp, and Fig. 7 b shows the sum of the probabilities of being in the RALp and AReLp states, as well as the individual probabilities of being in these states.

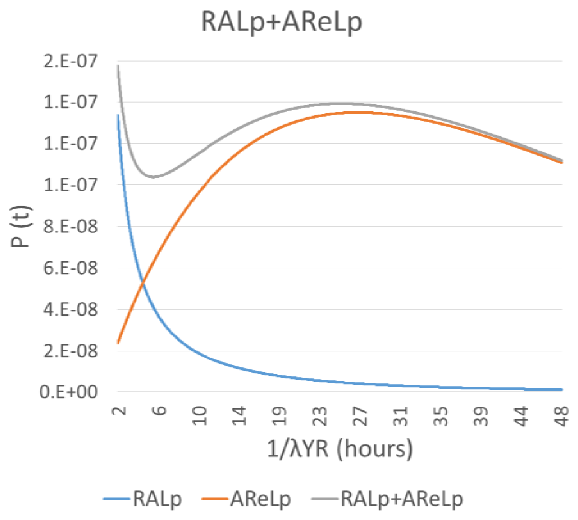




a b

Figure 6 – Calculations results of the model which doesn’t take into account the battery charge:
a – the probability of state AY; b – the total probability of states RA and ARe





a b

Figure 7 – Calculations results of the model which considers the battery charge:
a – the probability of AYSp state; b – the total probability of RALp and AReLp states

© Yakovyna V. S., Uhrynovskyi B. V., 2021
DOI 10.15588/1607-3274-2021-4-13

Figure 8 shows the probability that the mobile device is in a discharged state.

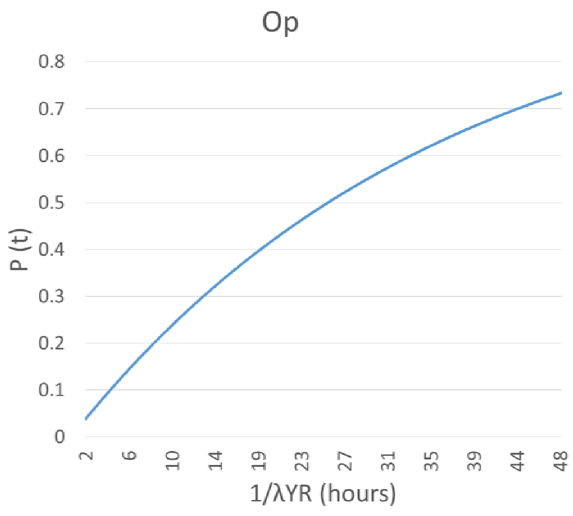


Figure 8 – The probability of state Op

6 DISCUSSION

Figures 6 (a) and 7 (a) show that as the time to the rejuvenation procedure increases, the probability of the system being in a young state decreases both considering the battery charge and without taking it into account.

The result of original aging model calculating (Fig. 6 (b)) shows that the optimal time to perform the rejuvenation procedure is 1 / λ_{yr} = 1620 minutes (27 hours), i.e., when the sum of the probabilities of being in the RA and ARe states is the lowest. In turn, the calculation of the model which considers the battery charge (Fig. 7 (b)) show that the rejuvenation procedure can be performed in 350 minutes (5 hours 50 minutes) after device start up. Comparing the results of calculations of both models, we can conclude that factor of battery charge has a significant impact on predicting the time of the rejuvenation procedure.

The idea of considering the battery charge factor is that performing a rejuvenation procedure is inefficient and impractical at a time when the battery level is critically low, or device is completely discharged. The formulated hypothesis is confirmed by the obtained calculations. If we do not consider the time to full discharge of the battery, the forecast of the original model is impractical, because after 27 hours (Fig. 8) the device is more likely to be discharged. The influence of the battery discharge can also be observed on Fig. 7 (b) where 1/ λ_{yr}>27.

It is worth noting that planning the rejuvenation procedure based on the battery level may not necessarily lead to performing this procedure at the selected time (5 hours 50 minutes), because at this time the system may still be in the state “Young”, and the battery may be recharged. Therefore, it is worth considering additional checks and new calculations of the model at scheduled time.

It is also necessary to note the reasons why the probabilities of being in the states “Recovering” and “Rebirth” are in the range of 10^{-7}. These two states are procedures that perform rebooting and restoring the system, so after performing them, the system immediately goes to the next state. At the same time, the probability of "Rebirth" is lower, because this state can be performed only if the mobile device is not used by the user and has a high battery charge. A high probability of being in these states would mean that the system is constantly or often in the process of rebooting, which is unacceptable for the user of the mobile device.

CONCLUSIONS

The paper describes software aging and rejuvenation model for mobile device which considers the factor of battery charge level.

The scientific novelty of the proposed mathematical model is to consider the new factor of battery charge level. The developed model is represented by a Continuous-Time Markov Chain, which combines the user behavior model, the model of software aging with the rejuvenation procedure, as well as the battery charge model.

The practical significance of using the battery level in the model to predict the rejuvenation time is to consider real mobile device usage scenarios, in particular, rejuvenation performing will not be effective or will not be performed at all in cases when the mobile device may have a very low charge or be completely discharged. In turn, early rejuvenation can delay the transition to a low battery level state, as rejuvenation will reduce the system workload and power consumption.

The future research is to investigate proposed model for different conditions and real data, because in this work a simulation was performed using test transition rates between states. Also, it is worth testing the hypothesis of the effect of the rejuvenation procedure performing before transition to the low battery state on the delay of the battery discharge by reducing the system workload.

ACKNOWLEDGEMENTS

The work is supported by the Ukrainian national state budget research project number 0121U109527.

REFERENCES

1. Parnas D. L. Software aging, *16th International Conference on Software Engineering, 1994, proceedings,* pp. 279–287. https://doi.org/10.1109/ICSE.1994.296790
2. Huang Y., Kintala C., Kolettis N., Fulton N. D. Software rejuvenation: Analysis, module and applications, *25th Symposium on Fault Tolerant Computing, 27–30 June 1995, proceedings*. Pasadena, California, 1995, pp. 381–390. https://doi.org/10.1109/FTCS.1995.466961
3. Grottke M., Jr R. M., Trivedi K. S. The fundamentals of software aging, *IEEE 19th International Symposium on Software Reliability Engineering : proceedings*, 2008, pp. 1–6.
4. Cotroneo D., Natella R., Pietrantuono R., Russo S. A Survey of Software Aging and Rejuvenation Studies, *ACM Journal on Emerging Technologies in Computing Systems,* 2014, Vol. 10, №1, Article 8. https://doi.org/10.1145/2539117
5. Valentim N. A., Macedo A., Matias R. A Systematic Mapping Review of the First 20 Years of Software Aging and Rejuvenation Research, *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW),* 2016. https://doi.org/10.1109/ISSREW.2016.42

6. Yakovyna V. S., Uhrynovskyi B. V. Software aging in the context of its reliability: a systematic review, *Scientific Bulletin of UNFU*, 2019, №29 (5), pp. 123–128. https://doi.org/10.15421/40290525
7. Ahamad S. Study of software aging issues and prevention solutions, *The International Journal of Computer Science and Information Security*, 2016, Vol. 14, No. 8, pp. 307–313.
8. Maji A. K., Hao K., Sultana S., Bagchi S. Characterizing failures in mobile OSes: a case study with Android and Symbian, *International Symposium on Software Reliability Engineering. IEEE Computer Society,* 2010, pp. 249–258. https://doi.org/10.1109/ISSRE.2010.45
9. Cotroneo D., Fucci F., Iannillo A. K., Natella R., Pietrantuono R. Software aging analysis of the android mobile os, *IEEE 27th International Symposium on Software Reliability Engineering*, 2016, pp. 478–489. https://doi.org/10.1109/ISSRE.2016.25
10. Yakovyna V.S., Uhrynovskyi B. V. Software aging in the context of reliability: a review of the issue, *Scientific Bulletin of UNFU,* 2020, No. 30 (2), pp. 107–112. https://doi.org/10.36930/40300219
11. Smartphone OS market share [Electronic resource]. Access mode: https://www.idc.com/promo/smartphone-market-share/os
12. Wu H., Wolter K. Software aging in mobile devices: Partial computation offloading as a solution, *IEEE International Symposium on Software Reliability Engineering Workshops*, 2015, pp. 125–131. https://doi.org/10.1109/ISSREW.2015.7392057
13. Weng C. Zhao D., Lu L., Xiang J., Yang C., Li D. A Rejuvenation Strategy in Android, *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW),* 2017, pp. 273–279. https://doi.org/10.1109/ISSREW.2017.50
14. Guo C., Wu H., Hua X., Lautner D., Ren S. Use Two-Level Rejuvenation to Combat Software Aging and Maximize Average Resource Performance, *IEEE 12th International Conference on Embedded Software and Systems*, 2015, pp. 1160–1165.
15. Xianga J., Wenga C., Zhaoa D., Tiana J., Xionga S., Lia L., A. Andrzejak A New Software Rejuvenation Model for Android, *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2018*. https://doi.org/10.1109/ISSREW.2018.00021
16. Stochastic Petri net [Electronic resource]. Access mode: https://en.wikipedia.org/wiki/Stochastic_Petri_net
17. Norris J. R. Markov Chains, Cambridge University Press, 1997.

Received 30.08.2021.
Accepted 16.10.2021.

УДК 004.93

МОДЕЛЬ ПРОЦЕСУ СТАРІННЯ ТА ОМОЛОДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID З УРАХУВАННЯМ ФАКТОРУ РІВНЯ ЗАРЯДУ БАТАРЕЇ

Яковина В. С. – д-р техн. наук, професор, професор кафедри Систем штучного інтелекту, Національний університет «Львівська політехніка», Львів, Україна; Факультет математики та комп'ютерних наук, Вармінсько-Мазурський Університет в Ольштині, Польща.

Угриновський Б. В. – асистент кафедри Програмного забезпечення, Національний університет «Львівська політехніка», Львів, Україна.

АНОТАЦІЯ

Актуальність. Особливість мобільних систем полягає в їх залежності від рівня заряду батареї, що є важливим чинником під час планування різного роду процесів, зокрема виконання процедури омолодження програмного забезпечення для зменшення впливу ефектів старіння цього програмного забезпечення.

Мета роботи. Розроблення моделі процесу старіння та омолодження програмного забезпечення для операційної системи Android з урахуванням чинника рівня заряду батареї.

Метод. Запропоновано комплексну модель на основі ланцюга Маркова з неперервним часом, яка об'єднює модель старіння із виконанням процедури омолодження програмного забезпечення, модель використання мобільного пристрою користувачем та фактор рівня заряду батареї. Побудовано граф станів та переходів, що описує об'єднані моделі. На основі діаграми написано систему диференційних рівнянь, яку обчислено з допомогою методу Рунге-Кутти 4-го порядку. Оптимальний час виконання процедури омолодження можна визначити за умов, коли її виконання не заважатиме користувачу та буде виконуватись завчасно до настання можливого повного розряду батареї, тобто тоді, коли ймовірність перебування системи в цих станах є найнижчою для певного значення часу виконання процедури омолодження.

Результати. Виконано симуляція розробленої моделі для тестових значень інтенсивностей переходів. Врахування моделі рівня заряду батареї дозволяє уникнути планування виконання процедури омолодження в час, коли мобільний пристрій з великою ймовірністю може мати низький заряд чи бути повністю розрядженим.

Висновки. Розроблена модель на основі ланцюга Маркова дозволить виконувати прогнозування часу початку процедури омолодження програмного забезпечення, враховуючи як поведінку користувача, так і рівень заряду батареї, який може мати значний вплив на прогнозований час. Також, раннє виконання процедури омолодження може мати вплив на зменшення навантаження на систему та відтермінування розряду пристрою, що варто перевірити в подальших дослідженнях. Обґрунтовано доцільність і важливість врахування чинника рівня заряду батареї і необхідність подальшого дослідження розробленої моделі старіння та омолодження із урахуванням нового чинника.

КЛЮЧОВІ СЛОВА: старіння програмного забезпечення, омолодження програмного забезпечення, ланцюг Маркова, операційна система Android.

УДК 004.93

МОДЕЛЬ ПРОЦЕССА СТАРЕНИЯ И ОМОЛОЖЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID С УЧЕТОМ ФАКТОРА УРОВНЯ ЗАРЯДА БАТАРЕИ

Яковина В. С. – д-р техн. наук, профессор, профессор кафедры Систем искусственного интеллекта, Национальный университет «Львовская политехника», Львов, Украина; Факультет математики и компьютерных наук, Варминско-Мазурский Университет в Ольштыне, Польша.

DOI 10.15588/1607-3274-2021-4-13

Угриновський Б. В. – ассистент кафедры Программного обеспечения, Национальный университет «Львовская политехника», Львов, Украина.

АННОТАЦИЯ

Актуальность. Особенность мобильных систем заключается в их зависимости от уровня заряда батареи, что является важным фактором при планировании различного рода процессов, в частности выполнения процедуры омоложения программного обеспечения для уменьшения влияния эффектов старения программного обеспечения.

Цель. Разработка модели процесса старения и омоложения программного обеспечения для операционной системы Android с учетом фактора уровня заряда батареи.

Метод. Предложено комплексную модель на основе цепи Маркова с непрерывным временем, которая объединяет модель старения с выполнением процедуры омоложения программного обеспечения, модель использования мобильного устройства пользователем и фактор уровня заряда батареи. Построен граф состояний и переходов, который описывает объединенную модель. На основе графа написано систему дифференциальных уравнений, которую вычислено с помощью метода Рунге-Кутты 4-го порядка. Оптимальное время выполнения процедуры омоложения можно определить в условиях, когда ее выполнение не будет мешать пользователю и будет выполняться заблаговременно до наступления возможного полного разряда батареи, то есть тогда, когда вероятность нахождения системы в этих состояниях является самой низкой для определенного значения времени выполнения процедуры омоложения.

Результаты. Выполнено симуляцию разработанной модели для тестовых значений интенсивностей переходов. Учет уровня заряда батареи позволяет избежать планирования выполнения процедуры омоложения в то время, когда мобильное устройство с большой вероятностью может иметь низкий заряд или быть полностью разряженным.

Выводы. Разработанная модель на основе цепи Маркова позволяет выполнять прогнозирования времени начала процедуры омоложения программного обеспечения, учитывая как поведение пользователя, так и уровень заряда батареи, который может оказать значительное влияние на прогнозируемое время. Также, раннее выполнение процедуры омоложения может влиять на уменьшение нагрузки на систему и отсрочку разряда устройства, что стоит проверить в дальнейших исследованиях. Обоснована целесообразность и важность учета фактора уровня заряда батареи и необходимость дальнейшего исследования разработанной модели старения и омоложения программного обеспечения с учетом нового фактора.

КЛЮЧЕВЫЕ СЛОВА: старение программного обеспечения, омоложение программного обеспечения, цепь Маркова, операционная система Android.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Parnas D. L. Software aging / D. L. Parnas // 16th International Conference on Software Engineering, 1994: proceedings. – P.279–287. https://doi.org/10.1109/ICSE.1994.296790
2. Software rejuvenation: Analysis, module and applications / [Y. Huang, C. Kintala, N. Kolettis, N. D. Fulton] // 25th Symposium on Fault Tolerant Computing, 27–30 June 1995: proceedings. – Pasadena, California, 1995. – P. 381–390. https://doi.org/10.1109/FTCS.1995.466961
3. Grottke M. The fundamentals of software aging / M. Grottke, R. M. Jr, K. S. Trivedi // IEEE 19th International Symposium on Software Reliability Engineering : proceedings. – 2008. – P.1–6.
4. Cotroneo D. A Survey of Software Aging and Rejuvenation Studies / D. Cotroneo, R. Natella, R. Pietrantuono, S. Russo // ACM Journal on Emerging Technologies in Computing Systems, 2014. – Vol. 10, № 1. – Article 8. https://doi.org/10.1145/2539117
5. Valentim N. A. A Systematic Mapping Review of the First 20 Years of Software Aging and Rejuvenation Research / N. A. Valentim, A. Macedo, R. Matias // IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2016. https://doi.org/10.1109/ISSREW.2016.42
6. Яковина В.С. Старіння програмного забезпечення в контексті його надійності: огляд проблематики / В. С. Яковина, Б. В. Угриновський // Науковий вісник НЛТУ України. – 2019. – №29 (5). – С. 123–128. https://doi.org/10.15421/40290525
7. Ahamad S. Study of software aging issues and prevention solutions / S. Ahamad // The International Journal of Computer Science and Information Security. – 2016. – Vol. 14, № 8. – P. 307–313.
8. Characterizing failures in mobile OSes: a case study with Android and Symbian / [A. K. Maji, K. Hao, S. Sultana, S. Bagchi] // International Symposium on Software Reliability Engineering. IEEE Computer Society. – 2010. – P. 249–258. https://doi.org/10.1109/ISSRE.2010.45
9. Software aging analysis of the android mobile os / [D. Cotroneo, F. Fucci, A. K. Iannillo et al.] // IEEE 27th International Symposium on Software Reliability Engineering. – 2016. – P. 478–489. https://doi.org/10.1109/ISSRE.2016.25
10. Яковина В. С. Старіння програмного забезпечення мобільних додатків: аналіз проблематики / В. С. Яковина, Б. В. Угриновський // Науковий вісник НЛТУ України. – 2020. – №30 (2). – С. 107–112. https://doi.org/10.36930/40300219
11. Smartphone OS market share [Electronic resource]. – Access mode: https://www.idc.com/promo/smartphone-market-share/os
12. Wu H. Software aging in mobile devices: Partial computation offloading as a solution / H. Wu, K. Wolter // IEEE International Symposium on Software Reliability Engineering Workshops. – 2015. – P. 125–131. https://doi.org/10.1109/ISSREW.2015.7392057
13. Weng C. A Rejuvenation Strategy in Android / [C. Weng, D. Zhao, L. Lu et al.] // IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). – 2017. – P. 273–279. https://doi.org/10.1109/ISSREW.2017.50
14. Guo C. Use Two-Level Rejuvenation to Combat Software Aging and Maximize Average Resource Performance / [C. Guo, H. Wu, X. Hua et al.] // IEEE 12th International Conference on Embedded Software and Systems. – 2015. – P. 1160–1165.
15. Xianga J. A New Software Rejuvenation Model for Android / [J. Xianga, C. Wenga, D. Zhaoa et al.] // IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). – 2018. https://doi.org/10.1109/ISSREW.2018.00021
16. Stochastic Petri net [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Stochastic_Petri_net
17. Norris J. R. Markov Chains / J. R. Norris // Cambridge University Press. – 1997.