Availability Model of Critical NPP I&C Systems with K-phase Erlang Distribution of Software Update

BogdanVolochiy[1], Vitaliy Yakovyna[1], Oleksandr Mulyak[1], Vyacheslav Kharchenko[2]

[1]National University Lviv Polytechnic, Lviv, Ukraine
bvolochiy@ukr.net, vitaliy.s.yakovyna@lpnu.ua,
mulyak.oleksandr@gmail.com
[2]National Aerospace University "KhAI", Kharkiv, Ukraine
v.kharchenko@csn.khai.edu

Abstract. This paper is the continuation of the research devoted to enhancing the adequacy of reliability model of Nuclear Power Plant (NPP) Instrumentation and Control (I&C) Systems considering software reliability. The reliability model of NPP I&C systems is a basement from which the availability, safety, risk, and other important characteristics of the system could be assessed. The availability function of a critical NPP I&C system depends on the hardware and software reliability and maintenance. The high availability value of the critical I&C systems could be ensured by following: structural redundancy; maintenance of the system; using the N-version programming; software updates. Thus the NPP I&C system reliability model to ensure its high level of adequacy and applicability has to take into account both software and hardware failures, as well as realistic distribution of software fault correction and K-phase Erlang Distribution of software updates are specified.

Keywords: Instrumentation and Control (I&C) System, Discrete-Continuous Stochastic System, Reliability Behavior, Structural-Automated Model, Markovian Chains, Software Reliability, Erlang distribution.

Key Terms. Mathematical Modeling, Method, Software Systems.

1 Introduction

1.1 Motivation

Nowadays the development of fault-tolerant computer-based systems (FTCSs) is a part of weaponry components, space, aviation, energy and other critical systems. One of the main tasks is to provide requirements of reliability, availability and functional safety. Thus the two types of possible risks relate to the assessment of risk, and to ensuring their safety and security.

Reliability (dependability) related design (RRD) [1-6] is a main part of development of complex fault-tolerant systems based on computers, software (SW) and hardware (HW) components. The goal of RRD is to develop the structure of FTCS tolerating HW physical failure and SW designs faults and assure required values of reliability, availability and other dependability attributes. To ensure fault-tolerance software, two or more versions of software (developed by different developers, using other languages and technologies, etc) are used [7].Therefore use of structural redundancy for FTCS with multiple versions of software is mandatory. When commissioning software some bugs (design faults) remain in its code [8], this leads to the shutdown of the FTCS. After detection the bugs, a software update is carried out. These factors have influence on the availability of the FTCS and should be taken into account in the availability indexes. The efficiency of fault-tolerant hardware of FTCS is provided by maintenance and repair.

Significant impact on the availability model of FTCS as discrete-continuous stochastic Markov systems has a real distribution of software debugging time. For that approach, there is an important problem of improving models by considering realistic distribution of procedure durations and time intervals between events in the process. Presented improvement allows to automate the usage of K-phase Erlang Distribution for the Markovian chain development of the statistical representation of the process of FTCS exploitation.

Insufficient level of adequacy of the availability models of FTCS leads either to additional costs (while underestimating of the measures), or to the risk of total failure (when inflating their values), namely accidents, material damage and even loss of life. Reliability and safety are assured by using (selection and development) fault-tolerant structures at RRD of the FTCS, and identifying and implementing strategies for maintenance. Adoption of wrong decisions at this stage leads to similar risks.

1.2 Related Works Analysis

Research papers, which focus on RRD, consider models of the FTCS [8-13]. Most models are primarily developed to identify the impact of one the above-listed factors on reliability indexes. The rest of the factors are overlooked. Papers [4, 5] describe the reliability model of FTCS which illustrates separate HW and SW failures. Paper [6] offer reliability model of a fault-tolerant system, in which HW and SW failures are differentiated and after corrections in the program code the software failure rate is accounted for. Paper [8] describes the reliability model of the FTCS, which accounts for the software updates. In paper [10] the author outlines the relevance of the estimation of the reliability indexes of FTCS considering the failure of SW and recommends a method for their determination. Such reliability models of the FTCS produce analysis of its conditions under the failure of SW. This research suggests that $MTTF_{system}=MTTF_{software}$. Thus, it is possible to conclude that the author considers the HW of the FTCS as absolutely reliable. Such condition reduces the credibility of the result, especially when the reliability of the HW is commensurable to the reliability of the SW. Paper [11] presents the assessment of reliability parameters of FTCS through modeling behavior using Markovian chains, which account for multiple software

updates. Nevertheless there was no evidence of the quantitative assessments of the reliability measures of presented FTCS.

In paper [12], the authors propose a model of FTCS using Macro-Markovian chains, where the software failure rate, duration of software verification, failure rate and repair rate of HW are accounted for. The presented method of Macro-Markovian chains modeling [12, 13] is based on logical analysis and cannot be used for profound configurations of FTCS due to their complexity and high probability of the occurrence of mistakes. Also there is a discussion around the definition of requirements for operational verification of software of the space system, together with the research model of the object for availability evaluation and scenarios preference. It is noted that over the last ten years out of 27% of space devices failures, which were fatal or such that restricted their use, 6% were associated with HW failure and 21% with SW failure.

Software updates are necessary due to the fact that at the point of SW commissioning they may contain a number of undetected faults, which can lead to critical failures of the FTCS. Presence of HW faults relates to the complexity of the system, and failure to conduct overall testing, as such testing is time consuming and needs substation financial support. To predict the number of SW faults at the time of its commissioning various models can be used, one for example is Jelinski-Moranda [14].

The K-phase Erlang method is used for development the analytical stochastic models of functional and reliability behaviors of technical system. Usage of this method for reliability predictions of technical system is presented in paper [14].Usage of this method for functional behavior for queuing system is presented in researches [15; 16, pp. 137-144;14, pp. 136-143].

In monographs [15, pp. 10-11, pp. 36-37; 17,pp. 196] states that any real distributions of the random variable is possible to present by “mix” of Erlang distributions $p_i(t_v)$:

$$p(t_v) = \sum_{i=1}^{k} q_i \, p_i(t_v) \quad for \quad t_v \geq 0 \ ,$$

where q_i – weighting coefficients, defining the share each of the Erlang distribution $p_i(t_v)$ which was used for presenting the real distribution.

A goal of the paper is to suggest structural-automated model to develop a Markovian chain for critical NPP I&C system with different redundancy types (first of all, structure and version) and real distribution of software update time, using the proposed formal procedure and tool. The main idea is to decrease risks of errors during development of Markovian chain (MC) for systems with very large (tens and hundreds) number of states. We propose a special notation which allows supporting development chain step by step and designing final MC using software tools. The paper is structured in the following way. The aim of this research is calculating the availability function of critical NPP I&C system with version-structural redundancy and double software updates.

To achieve this goal we propose a newly designed reliability model of critical NPP I&C system. As an example a special critical NPP I&C system is researched (Fig.2). The following factors are accounted for in this model: overall reserve of critical NPP

I&C system and joint cold redundancy of modules of main and diverse systems of critical NPP I&C system; the existence of three software versions; SW double update; physicals fault.

Structure of the paper is the following. Researched critical NPP I&C system is described in the second section. An approach to developing mathematical model based on Markovian chain and detailed procedure for the critical NPP I&C system are suggested in the third and fourth sections correspondingly. Simulation results for researched Markov's model are analyzed in the section 5. Last section concludes the paper and presents some directions of future researches and developments.

2 The Dependency of Software Failure Rates on the Frequence of Data Input Changing. the Randomness of Software Failures

Hardware ageing occurs naturally over time therefore, time is a generally accepted variable in functions for estimating hardware reliability. On the other hand, software failure will never occur if the software is not run. Therefore, in the context of software reliability, it is more practical to represent software run time as the number of computational operations completed by the software. It is important that the test metrics and ways of running the software take into account the following: that statistics of software failures should be investigated when a number of tasks are performing with different input data. Performing a task repeated times with the same input data will not result in a software failure if the software did not failure the first time the task was completed. Therefore, according to [18], the number and nature of software failures is the result of internal defects and depends on the conditions in which the software is used.

In contrast to hardware failures which can occur at any time regardless of input data, software failure depends on the frequency with which data are input. In practice, software can be considered such as some function *f*, which converts entry space into output space. The entry space is a set of all input conditions and the output space is the set of all output conditions [19]. Respective states are determined by a set of variables or typical software commands/transactions. According to standard IEC 60880, the input space which includes the branch of software performance is called the signal path [20]. In the period between two sets of incoming data being input, a software failure cannot occur if the software preliminary entry is performed and the software is in standby mode. Based on this, if the probability of software failure is calculated as10^4per 10,000 software launches and the frequency of input data from sensors is one second, the resulting software failure rate is calculated as 10^4 $seconds^{-1}$ or 0.36 $hours^{-1}$ (without taking into account the time required to create the output state), what can be unacceptable according to system reliability requirements, part of which is such software.

Failure is a concept in reliability theory and can be defined as an event, after the occurrence of which the characteristics of a technical object are outside of defined bounds [21]. According to [22], software failures are an event during which a soft-

ware failure is detected. The signs of software disability are outlined in technical documentation. The unrealistic results of software performance can be the result of temporary hardware failures or software defects. Software defects are the elements or parts of code, usage of which leads to unrealistic results [22]. In contrast to hardware failures, software failures can be caused by:

- incorrect algorithms or the incorrect implementation of algorithms ("write element of program");
- incorrect software documentation, which will lead to incorrect user actions;
- the input data which are being processed by the software;
- temporary failures of hardware which occur under external factors (ionizing radiation, temperatures, humidity or other factor), which can sometimes can be eliminated by a software restart.

3 The Use of Erlang Distributions for Software Updates in Availability Model of Fault-Tolerant Computer Based Systems with Version-Structural Redundancy

The duration of software updates is a random value and it is possible to present it by "mean time of new (next) software version development". The robustness of this measure should be founded in the resolution of decision reliability syntheses tasks of fault-tolerant computer based system which are an integral part of high availability critical infrastructure. The duration of new software version development depends on many factors such as: available staff with appropriated technology knowledge; qualification of developers; the complexity of the software and other factors. In previous papers [23, 24], it was assumed that the duration of software updates are a random variable with exponential distributions. However, it is more correct to consider software updates as a random variable with a normal (or Gaussian) distribution. If an availability model is developed in the discrete-continuous stochastic system form, the duration of all procedures in the fault-tolerant computer based system (included software updates) being analysed will be presented by exponential distributions. In this regard, the frequency with which software update durations occur in the vicinity of the mean is low compared to short software updates with a duration near zero. This leads to a decreasing adequacy of the availability model the object being studied.

This paper provides detailed suggestions for using Erlang distributions for the duration of software updates that will increase the adequacy of models and provide more realistic availability predictions. Figure 1 shows the probability density function for an exponential distribution and the probability density function of Erlang distributions with a shape parameter k=5. The probability of the occurrence of software update durations near the mean value in availability model with Erlang distributions is higher than when using exponential distributions for software updates. In a defined (given) interval, the area under the probability density function of the Erlang distribution (S1) is larger than are under the probability density function of the exponential distribution (S2). Difference between the probabilities of occurrences of software update durations would be increased by increasing the shape parameter k of the Erlang distributions.

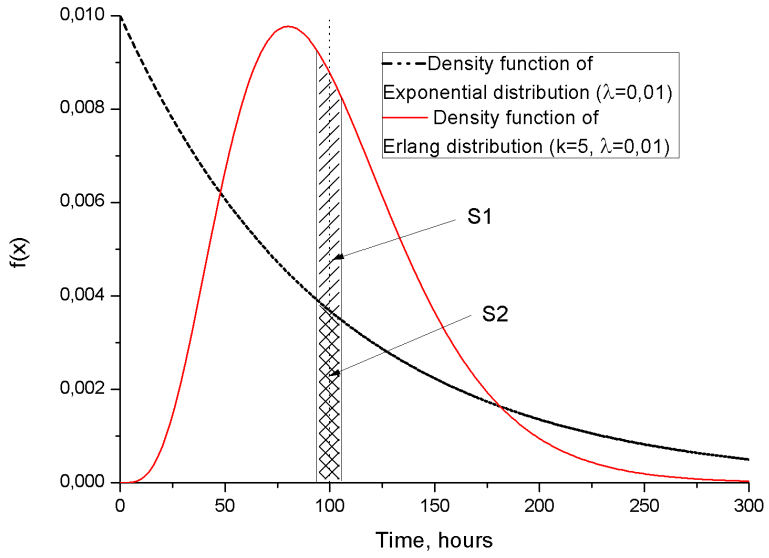

Fig. 1. Probability density function of Exponential and Erlang distributions with shape parameter equal 5

Based on this analysis, we conclude that Erlang distributions of software update durations is appropriate to use for availability models of fault-tolerant computer based systems in Markovian chain development. The main idea and instructions for implementing Erlang distributions with an arbitrary shape parameter (for any distributions) for the automated building of the Markovian chain was presented in paper [25].

4 Industry Case: FPGA Platform Based Reactor Trip System of NPP

Here we provide the structure (Fig.2) of researched safety critical NPP Instrumentation and Control system (I&C) based on the digital FPGA platform RadICS [26]. This is reactor trip system consisting of main and diverse systems [27]. Main and diverse systems have been developed using the FPGA safety controller (FSC) with three parallel channels on voting logic “2-out-of-3”.

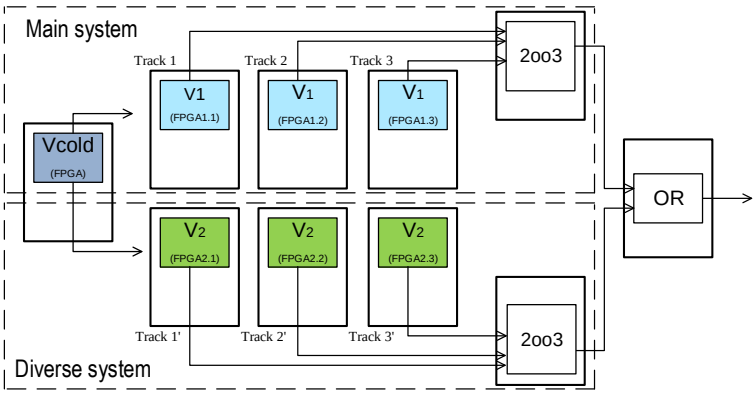

Fig. 2. Configuration of critical NPP I&C system

The output signals from main and diverse systems are binary (signals "switch-off" of reactor) and are joined according with logic OR (1 out of 2).

5 Markov's Model for Critical NPP I&C Systems with K-phase Erlang Distribution of Software Update

The method of formal and tool-based automated developing the Markovian chains for the researched critical NPP I&C systems are described in [9, 24]. It involves a formalized representation of the object of study as a "structural-automated model". The detailed availability model of critical NPP I&C systems was presented in our previous papers [23, 24]. In current paper we present modified structural-automated model considering the Erlang distributions of the software updates durations.

Structural-Automated Model of the critical NPP I&C systems for automated development of the Markovian chains are presented on the table 1. Improvements for Structural-Automated Model of consideration the K-phase Erlang Distribution of software debugging duration are presented below (all improvements are marked by bold font).

The parameters of the critical NPP I&C systems Markov's model: n – number of modules that are the part of the MS; k – number of modules that are the part of the DS; mc –number of the modules in the cold standby;λ_{hw}– the failure rate that is in MS or DS and in the hot standby; λ_{sw11}, λ_{sw12} – the failure rate of first and second software versions; K_e – number of Erlang Distribution phase; T_{up1}, T_{up2} – mean time of the first and second software updates; T_{switch} – mean time of the module connections from standby; T_{not} – mean time of developers notifications after software failures; T_{rep}– mean time of hardware repair.

Table 1. Structural-Automated Model of critical NPP I&C systems for the automated development of the Markovian chains

Terms and conditions of event	***Formula for calculating the rate of events***	***Rule of modification the state vectors component***
Event 1. Hardware failure of the MS module		
(V1>=(n-1)) AND (V6=0) AND (V10=0) **AND (V11=0) AND (V12=0)**	$V1 \cdot \lambda_{hw}$	V1:=V1-1; V8:=V8+1
Event 2. Software failure of the MS module		
(V1>=(n-1)) AND (V4=0) AND (V6=0) AND (V10=0) **AND (V11=0) AND (V12=0)**	λ_{sw11}	V4:=0; V6:=1
(V1>=(n-1)) AND (V4=1) AND (V6=0) AND (V10=0) **AND (V11=0) AND (V12=0)**	λ_{sw12}	V4:=1; V6:=1
Event 3. "Completing the developers notifications after software failures in main system		
(V1<=n) AND (V4=0) AND (V6=1) AND (V9=0) **AND (V11=0) AND (V12=0)**	$1/T_{not}$	V4:=0; V6:=1; V9:=1; **V11:=V11+1**
(V1<=n) AND (V4=1) AND (V6=1) AND (V9=0) **AND (V11=0) AND (V12=0)**	$1/T_{not}$	V4:=1; V6:=1; V9:=1; **V11:=V11+1**
Event 4. Hardware failure of diverse system module		
(V2>=(k-1)) AND (V7=0) AND (V9=0) AND (V11=0) AND (V12=0)	$V2 \cdot \lambda_{hw}$	V2:=V2-1; V8:=V8+1
Event 5. Software failure of the diverse system module		
(V2>=(k-1)) AND (V5=0) AND (V7=0) AND (V9=0) **AND (V11=0) AND (V12=0)**	λ_{sw11}	V5:=0; V7:=1
(V2>=(k-1)) AND (V5=1) AND (V7=0) AND (V9=0) **AND (V11=0) AND (V12=0)**	λ_{sw12}	V5:=1; V7:=1
Event 6. Completing the developers notifications after software failures in diverse system		
(V2<=k) AND (V5=0) AND (V7=1) AND (V10=0) **AND (V11=0) AND (V12=0)**	$1/T_{not}$	V5:=0; V7:=1; V10:=1; **V12:=V12+1**
(V2<=k) AND (V5=1) AND (V7=1) AND (V10=0) **AND (V11=0) AND (V12=0)**	$1/T_{not}$	V5:=1; V7:=1; V10:=1; **V12:=V12+1**
Event 7. Completing the procedure of software version updates in main system		
(V4=0) AND (V6=1) AND (V9=1) AND (V11>0) AND (V11<Ke)	$\mathbf{(1/T_{up1})\cdot P_{cor}}$	**V11:=V11+1**
(V4=0) AND (V6=1) AND (V9=1) **AND (V11=Ke)**	$(1/T_{up1})\cdot P_{cor}$	V4:=1; V6:=0; V9:=0; **V11:=0**
(V4=1) AND (V6=1) AND (V9=1) AND (V11>0) AND (V11<Ke)	$\mathbf{(1/T_{up2})\cdot P_{cor}}$	**V11:=V11+1**
(V4=1) AND (V6=1) AND (V9=1) **AND (V11=Ke)**	$(1/T_{up2})\cdot P_{cor}$	V4:=2; V6:=0; V9:=0; **V11:=0**

Terms and conditions of event	*Formula for calculating the rate of events*	*Rule of modification the state vectors component*
Event 8. Completing the procedure of software version updates in diverse system		
(V5=0) AND (V7=1) AND (V10=1) AND (V12>0) AND (V11<Ke)	**Ke($1/T_{up1}$)**	**V12:=V12+1**
(V5=0) AND (V7=1) AND (V10=1) **AND (V12=Ke)**	**Ke** ($1/T_{up1}$)	V5:=1; V6:=0; V10:=0; **V12:=0**
(V5=1) AND (V7=1) AND (V10=1) AND (V12>0) AND (V11<Ke)	**Ke($1/T_{up2}$)**	**V12:=V12+1**
(V5=1) AND (V7=1) AND (V10=1) **AND (V12=Ke)**	**Ke**($1/T_{up2}$)	V5:=2; V6:=0; V10:=0; **V12:=0**
Event 9. Completing the maintenances procedure of the system		
((V1<=n) OR (V2<=k)) AND (V8=4) **AND (V11=0) AND (V12=0)**	$1/T_{rep}$	V2:=k; V1:=n; V8:=0

The number of software updates can be also changed. It is necessary to change vectors V4 and V5 the ***event 7***, that are responsible for the number of updates. For example, if there are three software updates for diverse system, the entry component of the event will be as follows:

(V5=2) AND (V7=1) AND (V10=1) AND (V12>0) AND (V11<Ke)	Ke($1/T_{up3}$)	V12:=V12+1
(V5=2) AND (V7=1) AND (V10=1) AND (V12=Ke)	Ke($1/T_{up3}$)	**V5:=3;** V6:=0; V10:=0; V12:=0

The developed availability model of the critical NPP I&C system gives the possibilities according to technology [9] for automated construct of the Markovian chains. This construction provides a software module ASNA [28]. The Markovian chains which take into account the following settings critical NPP I&C system: n=3; k=3; m_c=0; λ_{hw}; λ_{sw11}, λ_{sw12}; T_{up1}, T_{up2}; T_{switch}; T_{rep}; K_e=0 are consists of 273 state and 893transitions. Information is available on the status of each software module ASNA we have on file "vector.vs", which is written in the form:

State 1: V1=3; V2=3; V3=0; V4=0; V5=0; V6=0; V7=0; V8=0; V9=0; V10=0; V11=0; V12=0
State 2: V1=2; V2=3; V3=0; V4=0; V5=0; V6=0; V7=0; V8=1; V9=0; V10=0; V11=0; V12=0
…….
State 481: V1=2; V2=1; V3=0; V4=1; V5=2; V6=1; V7=0; V8=3; V9=1; V10=0; V11=3; V12=0

For three phase parameters (K_e) of Erlang distribution for software updates durations were conducted tree Markovian chains with different dimension. The parameters of this Markovian chain in table 2 are shown.

Table 2. Parameters of Markovian chain with different phase number of Erlang distribution

n	k	K_e	Number of operation states	Number of non operation states	Number of fictitious states	Total number of states in model
3	3	0	120	153	0	**273**
3	3	3	120	153	208	**481**
3	3	5	120	153	416	**689**

The proposed structural-automated model of critical NPP I&C system for availability assessment can be easily transformed for other features of the object of study. It is enough to: add / remove basic event; attach / remove components of the state vector; and include / exclude parameters that describe the studied system. Based on information about the work of critical NPP I&C system an appropriate change in the model could be made (Fig. 2).

Based on the Markovian chains without Erlang distributions Ke=0 for critical NPP I&C system ("vector.vs") a system of differential equations was formed. Its solution allows us to estimate the function availability value of researched critical NPP I&C system. In the same manner the systems of differential equation for Markovian chain with Erlang distribution (Ke=3, Ke=5) of software updates was formed.

$$\begin{aligned}
\frac{dP_1(t)}{dt} &= -6\cdot(\lambda_{hw}+\lambda_{sw11})\cdot P_2(t)+\frac{1}{T_{repl}}\cdot\left(P_2(t)+P_3(t)+P_6(t)+P_7(t)+\right.\\
&\quad \left.+P_8(t)+P_9(t)+P_{11}(t)+P_{16}(t)\right)\\
\frac{dP_2(t)}{dt} &= -\frac{1}{T_{repl}}\cdot P_2(t)-2\cdot\lambda_{sw11}P_2(t)-2\cdot\lambda_{hw}\cdot P_2(t)-3\cdot\lambda_{hw}\cdot P_2(t)+\\
&\quad +2\cdot\lambda_{hw}P_1(t)\\
\frac{dP_3(t)}{dt} &= -\frac{1}{T_{repl}}\cdot P_3(t)-3\cdot(\lambda_{hw}+\lambda_{sw11})\cdot P_3(t)+2\cdot\lambda_{hw}P_2(t)\\
&\vdots\\
\frac{dP_{273}(t)}{dt} &= -\frac{1}{T_{repl}}\cdot P_{273}(t)+2\cdot\lambda_{hw}P_{156}(t)+2\cdot\lambda_{sw12}P_{272}(t)
\end{aligned}$$

Initial conditions for the system (2) is $P_1(t)=1;\ P_2(t)...P_{273}(t)=0.$

Based on the developed Markovian chains with different number of phase in Erlang distributions formulas for availability of critical NPP I&C system calculations could be assembled. Availability functions of critical NPP I&C system is calculated as the sum of the probability functions staying in operable states of chains. Based on Markovian chain of critical NPP I&C system availability function with different shape parameters are determined by following formulas:

$$A_{Ke=0}(t)=\sum_{i=1}^{12}P_i(t)+P_{14}(t)+\sum_{i=16}^{18}P_i(t)+\sum_{i=23}^{24}P_i(t)+\sum_{i=38}^{52}P_i(t)+\sum_{i=67}^{72}P_i(t)+\sum_{i=76}^{78}P_i(t)+$$
$$+P_{80}(t)+\sum_{i=82}^{83}P_i(t)+\sum_{i=87}^{88}P_i+\sum_{i=92}^{93}P_i+\sum_{i=122}^{144}P_i+\sum_{i=167}^{172}P_i+\sum_{i=176}^{177}P_i+P_{181}(t)$$
$$+\sum_{i=183}^{184}P_i+\sum_{i=186}^{187}P_i+\sum_{i=189}^{192}P_i+\sum_{i=215}^{230}P_i+\sum_{i=243}^{246}P_i+\sum_{i=251}^{254}P_i+\sum_{i=263}^{267}P_i+\sum_{i=271}^{272}P_i$$

$$A_{Ke=3}(t)=\sum_{i=1}^{12}P_i(t)+P_{14}(t)+\sum_{i=16}^{18}P_i(t)+\sum_{i=23}^{24}P_i(t)+\sum_{i=42}^{43}P_i(t)+\sum_{i=64}^{76}P_i(t)+\sum_{i=91}^{96}P_i(t)+P_{100}(t)+$$
$$+P_{102}(t)+P_{104}(t)+\sum_{i=106}^{108}P_i(t)+\sum_{i=111}^{112}P_i+\sum_{i=116}^{117}P_i+\sum_{i=121}^{122}P_i+\sum_{i=170}^{173}P_i+\sum_{i=178}^{179}P_i+$$
$$+\sum_{i=200}^{205}P_i+\sum_{i=226}^{236}P_i+\sum_{i=259}^{264}P_i+\sum_{i=268}^{269}P_i+P_{273}(t)+\sum_{i=275}^{279}P_i+\sum_{i=281}^{284}P_i+\sum_{i=328}^{330}P_i+\sum_{i=355}^{358}P_i$$
$$+\sum_{i=371}^{372}P_i+\sum_{i=393}^{398}P_i+\sum_{i=411}^{414}P_i+\sum_{i=419}^{422}P_i+\sum_{i=455}^{458}P_i+\sum_{i=471}^{472}P_i+\sum_{i=475}^{476}P_i$$

$$A_{Ke=5}(t)=\sum_{i=1}^{12}P_i(t)+P_{14}(t)+\sum_{i=16}^{18}P_i(t)+\sum_{i=23}^{24}P_i(t)+\sum_{i=46}^{47}P_i(t)+\sum_{i=88}^{100}P_i(t)+\sum_{i=115}^{120}P_i(t)+P_{124}(t)+$$
$$+P_{126}(t)+P_{128}(t)+\sum_{i=130}^{131}P_i(t)+\sum_{i=135}^{136}P_i+\sum_{i=140}^{141}P_i+\sum_{i=145}^{146}P_i+\sum_{i=218}^{221}P_i+\sum_{i=230}^{231}P_i+$$
$$+\sum_{i=272}^{277}P_i+\sum_{i=318}^{328}P_i+\sum_{i=351}^{356}P_i+\sum_{i=243}^{246}P_i+\sum_{i=360}^{361}P_i+P_{365}(t)+\sum_{i=367}^{368}P_i+\sum_{i=370}^{371}P_i+\sum_{i=373}^{376}P_i$$
$$+\sum_{i=439}^{442}P_i+\sum_{i=491}^{494}P_i+\sum_{i=519}^{520}P_i+\sum_{i=561}^{566}P_i+\sum_{i=579}^{582}P_i+\sum_{i=587}^{590}P_i+\sum_{i=647}^{650}P_i+\sum_{i=675}^{676}P_i+\sum_{i=679}^{680}P_i$$

6 Simulation Results

6.1 Research the Influence of K-phase Erlang Distribution of the Software Update Durations on Availability the Critical NPP I&C System

With the assistance of the proposed model, the following questions can be answered: What are the duration values of the first and the second software update (ensuring the values of the availability function of critical NPP I&C system of the initial phase of its operation do not reach below the specified level)? What are the allowed duration values of the first and the second SW updates? How the correlation between the first and the second SW does updates influence on the availability function? What is an influence on the availability function of duration of SW updates by Erlang distribution?

The experiment is conducted for the condition where the duration of the first software update is significantly shorter than the duration of the second update and the distribution of software update duration presented by Erlang distribution with different shape parameters K_e=0, K_e=3, K_e=5. The experiment is conducted with the fol-

lowing parameters critical NPP I&C system: $\lambda_{hw}=1\cdot10^{-5}$ hour^{-1}; $\lambda_{sw11}=2\cdot10^{-4}$hour^{-1}, $\lambda_{sw12}=1\cdot10^{-4}$hour^{-1}; T_{rep}=200 hour; T_{up1}=10 hour;T_{up2}=200 hours.

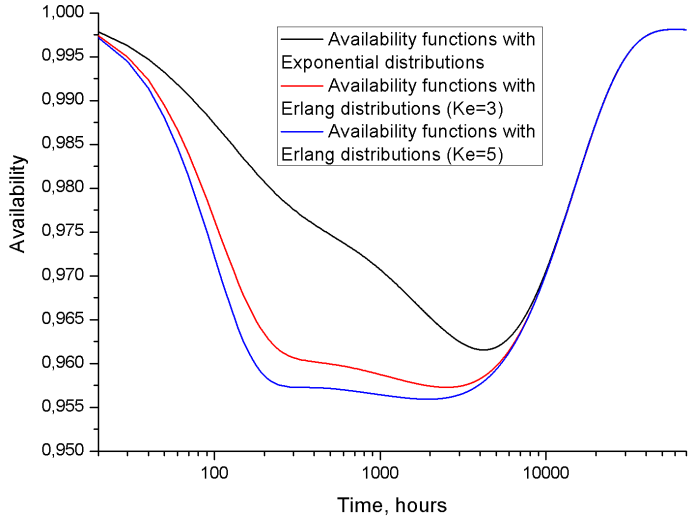

Fig. 3.Dependencies of availability function of the critical NPP I&C system on Erlang distributions (Ke=0, Ke=3, Ke=5) of software updates duration.

6.2 Analysis of the Results

The following results have been obtained by the proposed experiments:

1) Minimal levels of the availability functions for Markovian chains without Erlang distribution (Ke=0) for duration of software updates and Markovian chains with Erlang distributions (Ke=3, Ke=5) are changed. Analyzing the dependents of availability functions on the Fig. 3 could be concluded that real distribution of processes in system have significant effect of reliability indexes of critical NPP I&C system.

2) With the assistance of the proposed model it is possible to choose the duration of software updates that helps to ensure a minimum allowed level of the decrease of the availability function of the critical NPP I&C system.

3) In this research we confirmed that the exponential distribution in discrete-continuous stochastic models usage provides the limiting value of efficiency indicators. This occurrence was described by J. Martin in his monograph [pp.58-59, 29]. However, based on proposed model we confirmed that exponential distributions provide the most optimistic availability measures. The Erlang distribution for software updates usage provides a more realistic availability measure that is important during the operation of critical system and correction of software defects.

7 Conclusion

This research presents a model of critical NPP I&C system with double software updates and real distribution of software updates duration (by Erlang distributions presented) to illustrate automated development of Markovian chains using a special technology and tool ASNA.

The presented model can be easily adapted to different configurations of critical NPP I&C system, which envisages the use different majority voting, standby of the hardware part and as a consequence in the majority of software versions from different developers. In fact, this model can be adopted for an arbitrary number of software updates.

Future research has the potential to supplement this model with further factors: Erlang distribution for durations of hardware repair; unsuccessful restarting; unreliable commutation of elements and so on.

References

1. Mudry, P.A., Vannel, F., Tempesti, G., Mange, D. A reconfigurable hardware platform for prototyping cellular architectures. In: International Parallel and Distributed Processing Symposium. IEEE International, pp. 96--103 (2007)
2. Viktorov, O. Reconfigurable Multiprocessor System Reliability Estimation. Asian Jounal of Information Technology 6 (9), pp. 958--960(2007)
3. Rajesh, S., Vinoth Kumar C., Srivatsan, R., Harini, S., Shanthi, A. Fault Tolerance in Multicore Processors With Reconfigurable Hardware Unit. In: 15thInternational conference on high performance computing. Bangalore, INDIA, pp. 166--171 (2008)
4. Amerijckx, C., Legat, J.-D. A Low-Power Multiprocessor ArchitectureFor Embedded Reconfigurable Systems. In: Power and Timing Modeling, Optimization and Simulation, International Workshop, pp. 83--93 (2008)
5. Changyun Zhu, Gu, Z., Dick, R., Shang, L. Reliable multiprocessor system-on-chip synthesis. In: International Conference Hardware/Software Co-design and System Synthesis, pp. 239--244 (2007)
6. Kim P. Gostelow. The design of a fault-tolerant, realtime, multi-core computer system. In: In Aerospace Conference, IEEE, pp. 1--8(2011)
7. Lyu M.R. (ed.), Software Fault Tolerance, New York: John Wiley & Sons (1995)
8. Korotun, T.M. Models and methods for testing software systems. Programming problems, vol. 2, pp. 76--84 (2007) (In Russian)
9. Volochii, B.: Technology of modeling the information systems. Publishing NU "Lviv Polytechnic" (2004) (In Ukrainian)
10. Lei Xiong, Qingping Tan, Jianjun Xu. Effects of Soft Error to System Reliability. In: Workshops of International Conference on Advanced Information Networking and Applications. pp. 204--209 (2011)
11. Ponochonvyi, J.L., Odarushchenko, E.B. The reliability modeling non-redundant information and control systems with software updated. In: Radioelectronic and Computer Systems, vol. 4 (8), pp. 93--97 (2004) (In Russian)

12. Kharchenko, V., Odarushchenko, O., Odarushchenko, V.Popov, P. Selecting Mathematical Software for Dependability Assessment of Computer Systems Described by Stiff Markov Chains.Proc. Int. Conf. ICTERI, pp. 146--162 (2013)
13. Kharchenko, V., Ponochovny, Y., Boyarchuk, A. Availability Assessment of Information and Control Systems with Online Software Update and Verification. In: Information and Communication Technologies in Education, Research, and Industrial Applications Communications in Computer and Information Science, Vol. 469,Springer International Publishing Switzerland, pp. 300--324 (2014)
14. David R. Cox: Renewal theory. Methuen, 142 p. (1962)
15. Konig, D., Shtojan, D.: Methoden der Bedienungstheoric [Methods of Queueing Theory]. Vieweg, Braunschweig, 128 p. (1976) (In German)
16. Klejnrok, L:. Queueing systems. Volume I: Theory. John Wiley, New York, 432 p. (1976)
17. Rajnshke, K., Ushakov, I.A.: Assessment of The Reliability of The Systems Via Graphs. Moskva, Radio i svjaz' Publ., 208 p. (1988) (In Russian)
18. Lipaev V.: Software reliability. Moskov, Synteg, 232 p. (1998) (In Russian)
19. Hoang Pham: System Software Reliability. Springer Series in Reliability Engineering, 387 p. (2006)
20. IEC 60880. Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions, 217 p., (2006)
21. Polovko A.: Fundamentals of reliability theory of Saint Petersburg, 704 p. (2006) (In Russian)
22. DSTU 2844-94. Computer Software. Quality ensuring. Terms and definitions. Federal standart, 19 p. (1996) (In Ukrainian)
23. Volochiy, B., Mulyak, O., Kharchenko, V.: Automated Development of Markovian Chains for Fault Tolerant Computer-Based Systems with Version Structure Redundancy. Proceedings of the 11th International Conference on ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer, Vol. 1356. pp. 462--475 (2015)
24. Bogdan Volochiy, Oleksandr Mulyak, Leonid Ozirkovskyi, Vyacheslav Kharchenko: Automation of Quantitative Requirements Determination to Software Reliability of Safety Critical NPP I&C systems". In Proceedings of the Second International Symposium on Stochastic Models in Reliability Engineering, Life Science and Operations Management (SMRLO'16), pp. 337--346 (2016)
25. Fedasyuk, D.V., Volochiy, S.B.: Structural-automaton model of fault-tolerant systems for automated usage of erlang distribution Radioelektronic and Computer system, Vol. 3(77), Kharkiv, pp. 78—92, (2016) (In Ukrainian)
26. Kharchenko, V., Sklyar, V., Volkoviy, A.: Development and Verification of Dependable Multi-Version Systems on the Basic of IP-Cores. Proc. Int. Conf. Dependability of Computer Systems (2008)
27. Review Guidelines for Field-Programmable Gate Arrays in Nuclear Power Plant Safety Systems. NUREG/CR-7006, U.S. Nuclear Regulatory Commission, Washington, D.C., USA (2010)
28. Bobalo, J., Volochiy. B., Lozynskyi, O., Mandzii, B., Ozirkovskyi, L., Fedasuk, D., Shcherbovskyh, S., Jakovyna, V.: Mathematical models and methods for reliability analysis of electronic, electrical and software systems, Lviv Polytechnic Press, 425p, (2013) (In Ukrainian)
29. Martin J.: System Analysis for Data Transmission. IBM System Research Institute, Prentice Hall, Inc., Englewood Cliffs, 823p (1972)