Automated Development of Markovian Chains for Fault-Tolerant Computer-Based Systems with Version-Structure Redundancy

BogdanVolochiy[1], Oleksandr Mulyak[2], Vyacheslav Kharchenko[3]

[1]National University Lviv Polytechnic, Lviv, Ukraine
bvolochiy@ukr.net
[2]RPC "PromTechnoServis Ukraine", Kyiv, Ukraine
mulyak@prom-technoservice.com
[3] National Aerospace University "KhAI", Kharkiv, Ukraine
v.kharchenko@khai.edu

Abstract. Reliability design of fault-tolerant computer-based systems with version-structural redundancy and multiply software updates involves solving number of issues. This paper outlines an availability model of the computer-based systems which shows the algorithm for reliability behavior. For various configurations of the computer-based systems, the use of the proposed model and problem-oriented software, ASNA represents the ability to automate constructed the Markovian chains. This model includes a number of settings: failure rate of the software; numbers of software updates; duration of software updates; the structure of the system's hardware and reliability indicators. The proposed model for the automated development of Markovian chains is subject to the adaptation of the structure of the hardware of computer-based systems and/or the algorithms of reliability behavior. This allows us to obtain a new model and the feasibility to automate development of the Markovian chains.

Keywords. Markovian Chains, Automated Reliability Design, Fault-Tolerance System, Version-Structural Redundancy, Common Sliding Standby, Hot Standby, Cold Standby

Key Terms. Mathematical Modeling, Method, Software Systems.

1 Introduction

1.1 Motivation

Nowadays the developments of fault-tolerant computer-based systems (FTCSs) are 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 shut-down 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. During the operation of FTCS it is also possible that the HW will fail leading to failure of the software. To recover the software operability, an automatic restart procedure, which is time consuming, is performed. The efficiency of fault-tolerant hardware of FTCS is provided by maintenance and repair.

Insufficient level of adequacy of the availability models of FTCS leads either to additional costs (while underestimating of the indexes), 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. 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 modelling 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 modelling [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].

A goal of the paper is to suggest a technique to develop a Markovian chain for complex FTCS with different redundancy types (first of all, structure and version) using the proposed formal procedure and tool. The main idea is to decrease risks of errors during development of 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 FTCS with version-structural redundancy and double software updates.

To achieve this goal we propose a newly designed reliability model of FTCS. As an example a special computer-based system of space radio-technical complex is researched (Fig.1). The following factors are accounted for in this model: overall reserve of FTCS and joint sliding reserve of modules of main and diverse FTCS; the existence of two software versions; SW double update; and automatic software reboot, if its failure was caused by the HW physicals fault.

Structure of the paper is the following. Researched FTCS is described in the second section. An approach to developing mathematical model based on Markovian chain and detailed procedure for the FTCS 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 Researched fault-tolerant computer based system with structure-version redundancy

The researched FTCS with structure-version redundancy is shown on figure 1. To ensure the minimal FTCS downtime, overall hot standby with other version of software is used.

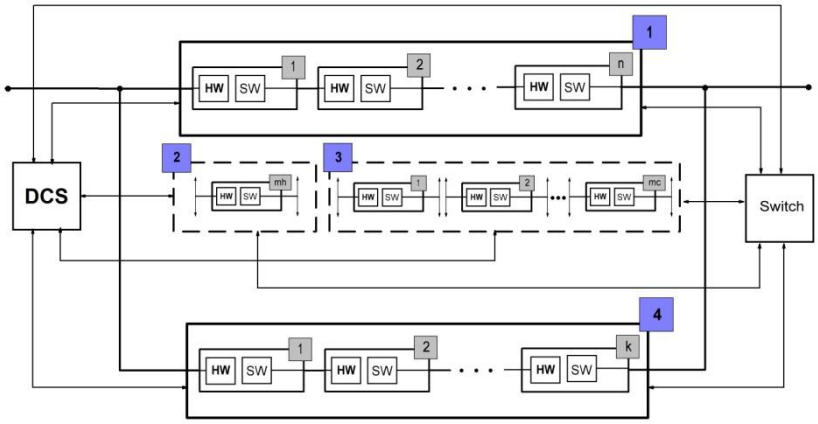

Fig. 1. Fault-tolerant Computer Based System (1 – main system, 2 – hot standby, 3 – cold standby, 4 – diverse system, DCS – Diagnostics Control System).

The FTCS consists of: a main system comprising modules; diverse system consist of k - modules; for two systems, the common sliding standby of modules is envisaged, the first module in hot standby and other in cold standby; a diagnostics control system determines the state of HW and SW, and manages the redundancy; and a switch is connected the modules to the main and diverse systems.

3 An approach to developing an availability model for FTCS with software update and restarting

An approach to the development of availability model for FTCS with double software updates and automatically software restart in the form of Markovian chains is presented in figure 2. During the operation of computer based system there are the following states: S1, S4 and S7 – system operable states; S2, S5, S9, S10 –inoperable states, in which SW updates, are conducted; S3, S6, S8 – inoperable states in which software restart after physical failure is automatically conducted; S11, S12, S13 – inoperable states in which HW is repaired after physical failure.

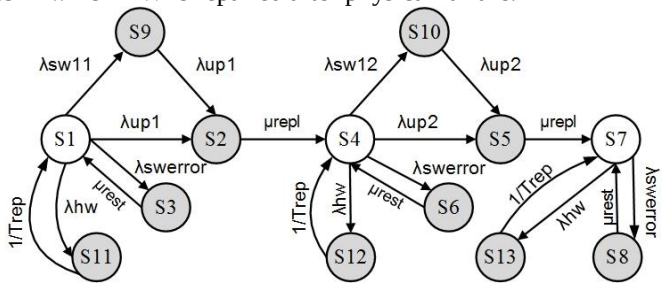

Fig. 2. Markovian chain which show the reliability behavior of computer-based system with double software updates and automatic restart

The system functioning after state S1 can unfold in four possible ways: the system moves to state S9 with rate λ_{sw11} after failure of first software version; the system move to state S3 with rate $\lambda_{swerror}$, after temporary failure of SW; the system move to state S2 with rate $\lambda_{up1}=1/T_{up1}$ (T_{up1} – duration of bugs correction in software) after finished the first version software operation (ready to use second version of software); the system move to state S11 with rate λ_{hw}, after physical failure.

The system functioning after state S4 can unfold in four possible ways: the system move to state S10 with rate λ_{sw12} after failure of second SW version; the system move to state S6 with rate $\lambda_{swerror}$, after temporary failure of SW; the system move to state S5 with rate $\lambda_{up2}=1/T_{up2}$ (T_{up2} – duration of bugs correction in software) after finished the second version SW operation (ready to use the third version of SW); the system move to state S12 with rate λ_{hw}, after physical failure. The system functioning after state S7 can unfold in two possible ways: the system move to state S6 with rate $\lambda_{swerror}$, after temporary failure of SW; the system move to state S13 with rate λ_{hw}, after physical failure. System moves from state S3and S6 to states S1 and S4 with a rate of $\mu_{rest}=1/T_{rest}$ (T_{rest} – duration of SW restart).

When the system is in state S2 and S5, it replaced the version of SW with rate $\mu_{repl}=1/T_{repl}$ (T_{repl} – duration of software replacement). The system moves from states S9 and S10 to states S2 and S5 with rate $\lambda_{up1}=1/T_{up1}$ (T_{up1} – duration of bugs correction in software) and $\lambda_{up2}=1/T_{up2}$ (T_{up2} – duration of bugs correction in software).

After commissioning of the computer based system starts debugging the software and develops first update that takes time T_{up1}. Second update takes timeT_{up2} and involves finding all bugs in SW. Therefore after first SW update, the numbers of bugs decreases, and debugging software is more complex, and the development of second update takes more time $T_{up2}>T_{up1}$.

The described approach to the development of availability model for FTCS with software update and restart is used to build availability model of FTCS showed in figure 1.

4 Markov's model for FTCS with software update and restarting

The method of development Markovian chain of the FTCS is described in the monograph [9]. It involves a formalized representation of the object of study as a "structural-automated model". To develop this availability model of the FTCS one needs to perform the following tasks: develop a verbal description of the research object (fig. 1); define the basic events; define the component vector of states, which can be described as a state of random time; define the parameters for the object of research, which should be in the model; and shape the tree of the modification of the rules and component of the vector of states.

4.1 The procedures to describe behavior of the FTCS

The FTTS behavior is described by the following procedures.

Procedure 1. Detection of failure of the FTCS (hardware failure, software failure, temporary failure). Failure can occur in the main and diverse system.

Procedure 2. Detection of failure in the main or diverse subsystems of the FTCS.

Procedure 3. Connection of the module from hot standby to faulty subsystem.

Procedure 4. Connection of the module from hot standby to cold standby.

Procedure 5. Loading the software on the module with connections from cold to hot standby.

Procedure 6. Software restart.

Procedure 7. Development the software updates.

Procedure 8. Repair (replacement) of the HW of the FTCS.

4.2 A set of the events for the FTCS

According to described procedures which determine the behavior of FTCS, a list of events is composed. Events are presented in pairs corresponding to the start and the end of time intervals to perform each procedure. From this list of events for "structural-automated model" basic events are selected [9].

As a result of analysis, twelve basic events in particular were determined: **Event 1** – "Hardware failure of main system module"; **Event 2** – "Software failure of the main system module"; **Event 3 –** "Software fault of the main system module"; **Event 4** – "Hardware failure of the diverse system module"; **Event 5 –** "Software failure of the diverse system module"**; Event 6 –** "Software fault of the diverse system module"; **Event 7 –** "Module failure in hot standby"; **Event 8 –** "Termination of the procedure of the hot standby module connection to non-operational system"; **Event 9 –** "Termination of the procedure of the cold standby module transfer to non-operational system"; **Event 10 –** "Termination of the procedure of software reloading on the module with failure feature in its software work"; **Event 11 –** "Termination of the procedure of SW version renovation"; **Event 12 –** "Termination of the procedure of the HW repair".

4.3 Components of vector states for the FTCS

Components of the vector state that can also be described as a state of random time. To describe the state of the system, eleven components are used: V1 – displays the current number of modules in the main system (the initial value of components V1 equal to n); V2 – displays the current number of modules in the diverse system (the initial value of components V2 equal to k); V3 – displays the current number of modules in hot standby (the initial value of components V3 equal to m_h); V4 – displays the current number of modules in cold standby (the initial value of components V4 equal to m_c); V5 – displays which software version is operated by the main system (V5=0 – first version, V5=1 – second version, V5=2 – third version); V6 – displays which software version operated by diverse system (V6=0 – first version, V6=1 – second version, V6=2 – third version); V7 – displays the temporary SW failure in the main system; V8 – displays the temporary SW failure in the diverse system; V9 – displays the SW fault in the main system; V10 – displays the SW fault in the diverse system; V11 – displays the number of non-operational units, due to HW fault.

4.4 The parameters of the FTCS Markov's model

Developing Markov's model of the FTCS, its composition and separate components should be set to relevant parameters in particular: n – number of modules that are the part of the main system; k – number of modules that are the part of the diverse system; m_h –number of the modules in the hot standby; m_c – number of the modules in the cold standby; λ_{hw} – the failure rate that is in main (diverse) system and in the hot standby; λ_{sw11}, λ_{sw12} – the failure rate of first and second software versions; $\lambda_{swerror}$ – the temporary failure rate of software; T_{up1}, T_{up2} – duration of the first and second software updates; T_{rest} – duration of software restart on the module; T_{switch} – duration of the module connections of the slight standby; T_{rep}– hardware repair duration.

4.5 Model of the FTCS for the automated development of the Markovian chain with software update and restart

According to the technology of analytical modeling, the discrete-continuous stochastic systems [9] based on certain events using the component vector state and the parameters that describe FTCS, and model of the FTCS for automated development of the Markovian chains are presented on the table 1.

Table 1. Model "Structural-Automated Model" of the FTCS for the automated development of the Markovian chains

Terms and conditions	**Formula used for the intensity of the events**	**Rule of modification component for the state vector**
Event 1. Hardware failure of main system module		
(V1=n) AND (V3>0) AND (V9=0)	$V1 \cdot \lambda_{hw}$	V1:=n; V3:=V3-1; V11:=V11+1
(V1=n) AND (V3=0) AND (V9=0)	$V1 \cdot \lambda_{hw}$	V1:=V1-1; V11:=V11+1
Event 2. Software failure of the main system module		
(V1=n) AND (V3=0) AND (V9=0)	$V1 \cdot \lambda_{swerror}$	V1:=V1-1; V7:=V7+1
(V1=n) AND (V3>0) AND (V9=0)	$V1 \cdot \lambda_{swerror}$	V1:=n; V3:=V3-1; V7:=V7+1
Event 3. Software fault of the main system module		
(V1=n) AND (V5=0) AND (V9=0)	$V1 \cdot \lambda_{sw11}$	V1:=V1-1; V5:=0; V9:=1
(V1=n) AND (V5=1) AND (V9=0)	$V1 \cdot \lambda_{sw12}$	V1:=V1-1; V5:=1; V9:=1
Event 4. Hardware failure of the diverse system module		
(V2=k) AND (V3>0) AND (V10=0)	$V2 \cdot \lambda_{hw}$	V2:=k; V3:=V3-1; V11:=V11+1
(V2=k) AND (V3=0) AND (V10=0)	$V2 \cdot \lambda_{hw}$	V2:=V2-1; V11:=V11+1
Event 5. Software failure of the diverse system module		
(V2=k) AND (V3=0) AND (V10=0)	$V2 \cdot \lambda_{swerror}$	V2:=V2-1; V8:=V8+1
(V2=k) AND (V3>0)	$V2 \cdot \lambda_{swerror}$	V2:=k; V3:=V3-1; V8:=V8+1
Event 6. Software fault of the diverse system module		

(V2=k) AND (V6=0) AND (V10=0)	$V2 \cdot \lambda_{sw11}$	V2:=V2-1; V6:=0; V10:=1
(V2=k) AND (V6=1) AND (V10=0)	$V2 \cdot \lambda_{sw12}$	V2:=V2-1; V6:=1; V10:=1
Event 7. Module failure that is in the hot standby		
(V3>0) AND ((V9=0) OR (V10=0))	$V3 \cdot \lambda_{hw}$	V3:=V3-1; V11:=V11+1
Event 8. Termination of the procedure of the hot standby module connection to non-operational system		
(V1<n) AND (V3>0) AND (V11>0)	$1/T_{switch}$	V1:=V1+1; V3:=V3-1
(V2<k) AND (V3>0) AND (V11>0)	$1/T_{switch}$	V2:=V2+1; V3:=V3-1
Event 9. Termination of the procedure of the cold standby module transfer to non-operational CS		
(V3<mh) AND (V4>0)	$1/T_{switch}$	V3:=V3+1; V4:=V4-1
Event 10. Termination of the procedure of software reloading on the module with failure feature in its software work		
(V1<n) AND (V7>0)	$1/T_{rest}$	V1:=V1+1; V7:=V7-1
(V2<k) AND (V8>0)	$1/T_{rest}$	V2:=V2+1; V8:=V8-1
Event 11. Termination of the procedure of software version renovation		
(V1<n) AND (V5=0) AND (V9=1)	$1/T_{up1}$	V1:=n; V5:=1; V9:=0
(V1<n) AND (V5=1) AND (V9=1)	$1/T_{up2}$	V1:=n; V5:=2; V9:=0
(V2<k) AND (V6=0) AND (V10=1)	$1/T_{up1}$	V2:=k; V6:=1; V10:=0
(V2<k) AND (V6=1) AND (V10=1)	$1/T_{up2}$	V2:=k; V6:=2; V10:=0
Event 12. Termination of the procedure of the hardware repair		
(V1<n) AND (V2<k) AND (V11=2)	$1/T_{rep}$	V1:=n; V2:=k; V11:=0

As shown in Table 1, the model can be easily adapted for other fault-tolerant hardware configurations, or any number of software updates can be provided. For example, if the main and diverse system built with rule of voting 2-out-of-3, the description of the ***terms and conditions*** for the ***events of 1-6*** should be replaced with (V1=n) → (V1<=2) i (V2=k) → (V2<=2). In this way, we obtain the FTCSC for the automated development of the Markovian chains in which permanent and diverse fault-tolerant systems are built with rule of voting 2-of-3.

The number of software updates can be also changed. It is necessary to change vectors V5 and V6 the ***event 11***, that are responsible for the number of updates. For example, if there are three software updates, the entry component of the event will be as follows:

(V1<n) AND **(V5=2)** AND (V9=1)	$\mathbf{1/T_{up3}}$	V1:=n; **V5:=3;** V9:=0
(V2<k) AND **(V6=2)** AND (V10=1)	$\mathbf{1/T_{up3}}$	V2:=k; **V6:=3;** V10:=0

In order to present the necessary strategy of hardware repair can be transformed ***the event 12***.

4.6 Automated development of the Markovian chain and determining of availability function

The developed availability model of the FTCS gives the possibilities according to technology [9] for automated construct of the Markovian chains. This construction provides a software module ASNA [15]. The Markovian chains which take into account the following settings FTCS: n=2; k=2;m_h=0; m_c=0; λ_{hw}; λ_{sw11}, λ_{sw12}; $\lambda_{swerror}$; T_{up1}, T_{up2}; T_{rest}; T_{switch}; T_{rep}, are presented in figure 3. 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=2; V2=2; V3=0; V4=0; V5=0; V6=0; V7=0; V8=0; V9=0; V10=0; V11=0
State 2: V1=1; V2=2; V3=0; V4=0; V5=0; V6=0; V7=0; V8=0; V9=0; V10=0; V11=1
…….
State 121: V1=1; V2=1; V3=0; V4=0; V5=2; V6=2; V7=1; V8=1; V9=0; V10=0; V11=0

As the configurations of researched FTCS changes, the dimension of graphs increases. Therefore for the configuration of FTCS (Fig. 1) with one module in a hot reserve graph has 395 states and 976 transitions.

The proposed availability model of FTCS can be easily transformed for other features of the object of study. It is enough to: add / remove basic event (4.2); attach / remove components of the state vector (4.3); and include / exclude parameters that describe the studied system (4.4). Based on information about the work of FTCS an appropriate change in the model could be made (Fig 1).

Basing on the Markovian chains (Fig 3) formulas for designing of availability FTCS can be assembled. One measure of the availability of recovered FTCS reveals it is an availability function. Availability function of FTCS is calculated as the sum of the probability functions staying in operable states of chains. Basing on these states the FTCS availability function with parameters of FTCS n=2; k=2; m_h=0; m_c=0 is determined by the formula (1):

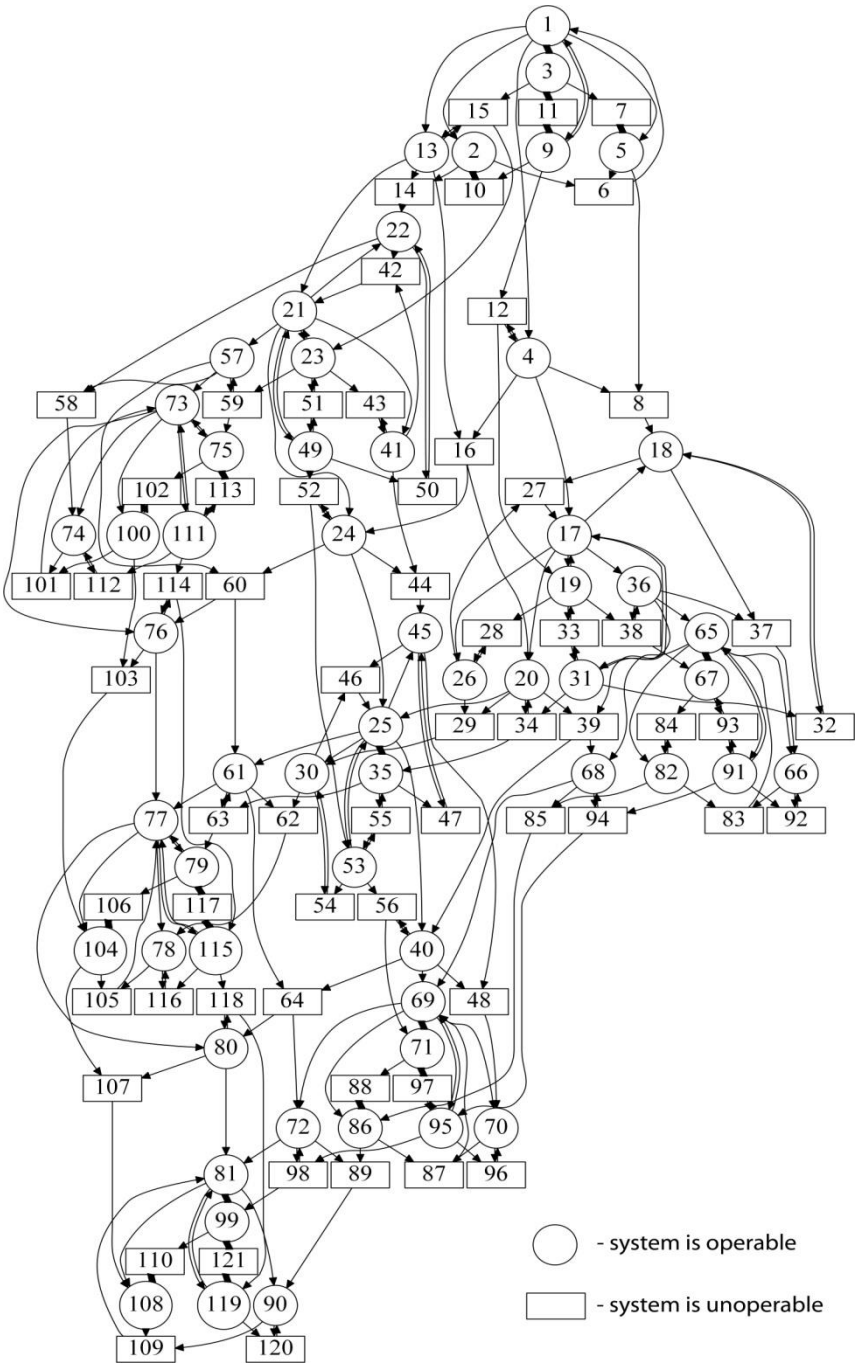

Fig. 3. The Markovian chains of the researched FTCS

$$K_{\Gamma}(t)=\sum_{i=1}^{5} P_i(t)+P_9(t)+P_{13}(t)+\sum_{i=17}^{26} P_i(t)+\sum_{i=30}^{31} P_i(t)+\sum_{i=33}^{36} P_i(t)+\sum_{i=40}^{41} P_i(t)+P_{45}(t)+P_{49}+$$
$$+(t)+P_{53}(t)+P_{57}(t)+P_{61}(t)+\sum_{i=65}^{82} P_i(t)+P_{86}(t)+\sum_{i=90}^{91} P_i(t)+P_{95}(t)+P_{95}(t)+\sum_{i=99}^{100} P_i(t)+ \quad (1)$$
$$+P_{104}(t)+P_{108}(t)+P_{111}(t)+P_{115}(t)+P_{119}(t)$$

Based on the Markovian chains (Fig.3) a system of differential equations (2) was formed. Its solution allows us to estimate the function availability value of researched FTCS.

$$\left.\begin{aligned}
&\frac{dP_1(t)}{dt}=-2\cdot\lambda_{hw}\left(P_2(t)+P_5(t)\right)-2\cdot\lambda_{sw11}\left(P_4(t)+P_{13}(t)\right)-2\cdot\lambda_{swerror}\cdot P_3(t)-\\
&\qquad -2\cdot\lambda_{swerror}\cdot P_9(t)+\frac{1}{T_{rep}}\cdot P_6(t)+\frac{1}{T_{rest}}\cdot\left(P_3(t)+P_3(t)\right)\\
&\frac{dP_2(t)}{dt}=2\cdot\lambda_{hw}P_1(t)-\frac{1}{T_{rest}}\cdot P_{10}(t)-2\cdot\lambda_{hw}P_6(t)-2\cdot\lambda_{swerror}\cdot P_{10}(t)-\\
&\qquad -2\cdot\lambda_{sw11}P_{14}(t)\\
&\frac{dP_3(t)}{dt}=2\cdot\lambda_{hw}P_1(t)+\frac{1}{T_{rest}}\cdot P_3(t)-2\cdot\lambda_{hw}P_7(t)-2\cdot\lambda_{swerror}\cdot P_{11}(t)-\\
&\qquad -\frac{1}{T_{rest}}\cdot P_{11}(t)-2\cdot\lambda_{sw11}P_{15}(t)\\
&\vdots\\
&\frac{dP_{121}(t)}{dt}=-\frac{1}{T_{rest}}\cdot P_{90}(t)+2\cdot\lambda_{swerror}\cdot P_{90}(t)+2\cdot\lambda_{hw}P_{119}(t)
\end{aligned}\right\} \quad (2)$$

Initial conditions for the system (2) is $P_1(t)=1;\ \ P_2(t)...P_{121}(t)=0.$

5 Simulation results

5.1 Research of influence of software updates duration on the availability function

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 FTCS 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 does the correlation between the first and the second SW updates influence on the availability function?

The first experiment is conducted for the condition where the duration of the first software update is significantly shorter than the duration of the second update. The duration of the first update is given within 10 - 50 hours, and the duration of the second update - 200 hours. The experiment is conducted with the following parameters FTCS: $\lambda_{hw}=1\cdot10^{-5}$ hour^{-1}; $\lambda_{sw11}=2\cdot10^{-3}$ hour^{-1}, $\lambda_{sw12}=1\cdot10^{-3}$ hour^{-1}; $\lambda_{swerror}=1\cdot10^{-2}$ hour^{-1}; T_{rest}=6 min; T_{switch}; T_{rep}=200 hour; T_{up2}=200 hour; (*line 1* -T_{up1}=10 hour; *line2* - T_{up1}=20 hour; *line3* - T_{up1}=30 hour; *line4* - T_{up1}=40 hour; *line5* -T_{up1}=50 hour).

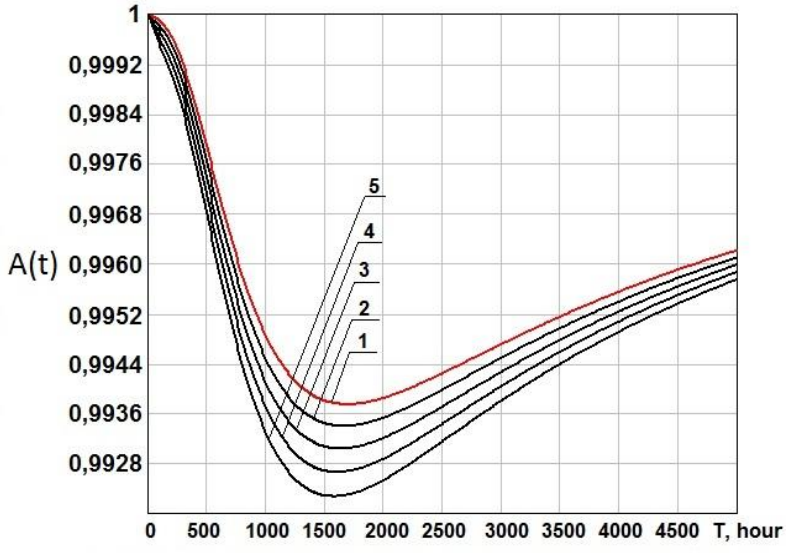

Fig. 4. Dependencies of availability function of the FTCS on values of the software update durations (duration of the first software update for 10 to 50 hours; the duration of the second firmware update - 200 hours).

The second experiment is conducted for the condition where the duration of the first SW update is significantly longer then the duration of the second SW update. The duration of the first update is 200 hours and the duration of the second update is given within 10 - 50 hours. The experiment is conducted with the following parameters FTCS: $\lambda_{hw}=1\cdot10^{-5}$ hour^{-1}; $\lambda_{sw11}=2\cdot10^{-3}$ hour^{-1}, $\lambda_{sw12}=1\cdot10^{-3}$ hour^{-1}; $\lambda_{swerror}=1\cdot10^{-2}$ hour^{-1}; T_{rest}=6 min; T_{switch}; T_{rep}=200 hour; T_{up1}=200 hour; (*line 1* - T_{up2}=10 hour; *line2* - T_{up2}=20 hour; *line3* - T_{up2}=30 hour; *line4* - T_{up2}=40 hour; *line5* - T_{up2}=50 hour).

The following results are produced by the proposed experiments:

1) If the duration of the first software update is significantly shorter than the duration of the second update, a decrease of the availability function of the readiness of the operational interval to 1700 hours is observed. If the duration of the first software update is significantly longer the decrease of the availability function, are readiness of the operational interval to 800 hours is observed.

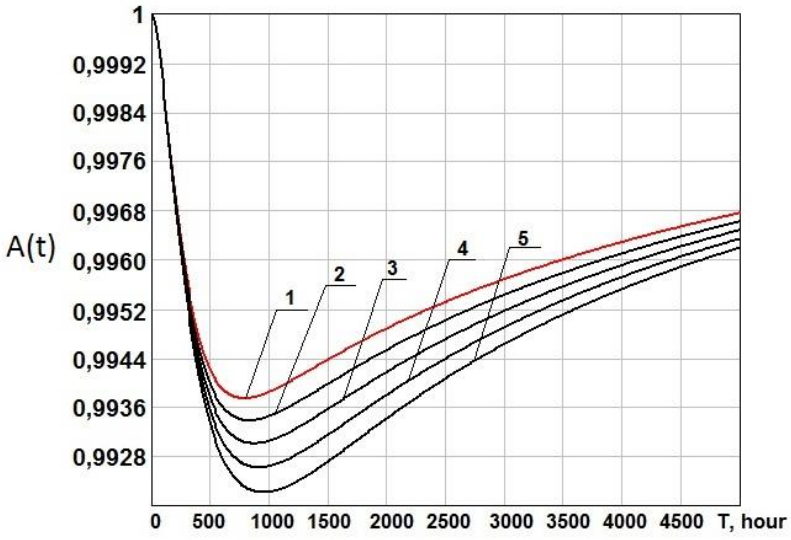

Fig. 5. Dependencies of availability function of the FTCS on the values of software update duration (duration of the first software update - 200 hours; the duration of the second firmware update from 10 to 50 hours).

2) The minimal decrease level of the availability function of the readiness of FTCS in the first and the second experiments is the same. This is explained by the values chosen for the duration for the first and the second software updates.

3) 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 FTCS.

6 Conclusion

This research presents a model of FTCS with version-structural redundancy, with software updates and restart for automated development of Markovian chains using a unique technology and specific tool ASNA.

The presented model can be easily adapted to different configurations of FTCS, which envisages the use of the majority voting reservation in the hardware part and as a consequence in the majority of software versions from different developers. This model can be adopted for an unlimited number of software updates.

Future research has the potential to supplement this model with further factors:

- Erlang distribution for durations of software updates [15];
- 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),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: 15th International conference on high performance computing. Bangalore, INDIA, pp. 166–171 (2008)
4. Amerijckx, C., Legat, J.-D. A Low-Power Multiprocessor Architecture For 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 Codesign 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 2, 76–84 (2007) (In Russian)
9. Volochii, B.: Technology of Modeling the Information Systems. Publishing NU "Lviv Polytechnic" (2004) (In Ukranian)
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. Radioelectronic and computer systems 4(8), 93–97 (2004) (In Russian)
12. 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)
13. Kharchenko V., Ponochovny Y., Boyarchuk A., Ermolayev V.: 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. Moranda, P. B. An Error Detection Model for Application During Software Development. IEEE Trans. Reliability 4, 309–312 (1981)
15. Bobalo, J., Volochiy. B., Lozynskyi, O., Mandzii, B., Ozirkovskyi, L., Fedasuk, D., Shcherbovskyh, S., Yakovyna, V. Mathematical Models and Methods for Reliability Analysis of Electronic, Electrical and Software Systems, Lviv Polytechnic Press (2013)