Assessment of the Reactor Trip System Dependability

Two Markov Chains - based Cases

Vyacheslav Kharchenko
Department of Computer Systems and Networks
National aerospace university "KhAI"[1]
Kharkiv, Ukraine
v.kharchenko@csn.khai.edu

Elena Odarushchneko
Poltava National Technical University
Department of Applied Mathematics, Informatics and Mathematical Modeling
skifs2007@mail.ru

Oleg Odarushchenko
Research and Production Corporation Radiy
Kirovograd, Ukraine
odarushchenko@gmail.com

Dmitriy Butenko
AltexSoft
Kharkiv, Ukraine
twilightwnd@gmail.com

Valentina Butenko[1]
v. odarushchenko@csn.khai.edu

***Abstract* — The diversity approach is commonly used to ensure dependability attributes of such critical computer-based systems as NPP I&C systems, including the Reactor Trip System. There are two basic problems emerging in this approach – the choice of product-process diversity kinds and assessment of multi-version systems dependability. This paper presents a study for dependability assessment of two architectures of a typical NPP I&C system, the Reactor Trip System: a diverse two-channel system with three parallel tracks on voting logic "2-out-of-3" in each channel and a diverse three-channel system with two parallel tracks on voting logic "1-out-of-2" in each channel. The multi-fragmentation approach is proposed to provide a detailed description of RTS hardware-software interconnection. The resulting models are solved using a number of approaches and tools to verify the results. Obtained results can help to make informed decision between the observed RTS architectures.**

Keywords—Reactor Trip Systems, Markov chains, multi-fargmentation, stiffness

I. INTRODUCTION

The diversity approach is commonly used to ensure reliability, availability, safety and other dependability attributes of critical computer-based systems, which implies the creation of several diverse design options for redundant channels or parts of the channels (hardware-software products) [1]. There are two basic problems emerging in this approach – the choice of product-process diversity kinds and assessment of multi-version systems dependability [2], [3].

Provision of an accurate dependability assessment of multi-version complex safety-critical systems, such as NPPs I&C system, is of high importance for the development process. Assessment errors may lead to either wrong or suboptimal decisions. Dependability parameters of such systems are assessed using probabilistic models. State-space methods, such as Markov chains (MC) [4] are preferred to reliability block diagrams (RBD) and fault trees as they can better handle such complex situations as failure/repair dependencies and shared repair resources [5]. One of the main computational difficulties of MC applications is the size of the model's state space.

System modelers are often interested in transient measures, which provide more useful information than steady-state measures. Modeling components interaction and interdependencies expands the model significantly, thus making the precise computation of system transient measures almost infeasible. Whilst numerical methods and imitation modeling can be applied to handling this problem, they are also limited by model size and such difficulties as stiffness [6] and sparsity [7]. Stiffness is a well-known undesirable property of many practical MCs [8] as it poses a problem of finding transient solutions. Using solution methods that do not preserve sparsity, is unacceptable for most large problems [7].

Several techniques were developed to deal efficiently with MC largeness and stiffness [6, 8, 9] and a number of tools can be used to find the MC solution [10, 12]. Such variety of tools and approaches is extremely helpful in the process of system modeling but this also poses a difficulty when it comes to choosing the most appropriate method for a specific assessment. As every tool is limited in its properties and applicability, a careful selection is needed for the tools used to solve large and stiff MCs accurately and efficiently. It should be noted that stiffness usually requires the modeler to focus on a number of math details to avoid the use of inefficient approaches, methods [12], and tools. This view goes against the recommendations of one of the leading standards in the safety area, IEC 61508-6 [13]. This standard asserts that methods for solving Markov models have been developed long ago and trying to improve these methods does not seem

sensible[1]. The previous works show [10] that solving a large and/or stiff Markov model requires a careful selection of the solution method/tool. Using an inappropriate method/tool for the solution of a non-trivial MC may lead to significant errors.

In this paper, we present the study for dependability assessment of a typical NPP instrumentation and control system (I&Cs), the Reactor Trip System (RTS) based on a digital platform produced by RPC Radiy. This is an FPGA-based system. We observe two possible RTS architectures: a two-channel system with three parallel tracks on voting logic "2-out-of-3" in each channel or a three-channel system with two parallel tracks on voting logic "1-out-of-2" in each channel. Each system uses diverse software versions. We use the multi-fragmentation (MFM) approach to model the system hardware-software interconnection and behavior, and a number of approaches and tools to find the transient solution and verify the results. The availability functions of both architectures are analyzed and compared. The results of such comparison can help to make an informed decision about the preferred architecture for a particular system. The paper consists of the following sections: Section 2 provides the general description of the multi-fragmentation approach, which was used to model both RTS architectures; Section 3 presents the description of the digital FPGA-based platform and an overview of RTS architectures; Section 4 presents the Markov models of those architectures taking physical faults into account; Section 5 presents the MFM for both architectures; Section 6 presents the solution results using several approaches and tools. In Section 7 we present the conclusions.

II. MULTI-FRAGMENTATION APPROACH

The main idea of the MFM approach is to present the complex hardware-software behavior of the researched system, taking into account the known residual amount of software faults. Using this approach the model is divided into *N* fragments that are with the same structure but may differ in one or more parameter values [21]. The model fragments describe the functioning of system hardware part and transitions between fragments are show the process of software failures detection.

The approach is based on the following assumptions.

- The number of software defects is a known finite value.
- The failure rate of design faults $\lambda_{d(i)}$ is proportional to their residual amount n_i in *i* different software versions [14]. This is a well-known assumption of concave software reliability grows models (SRGM) [17-18]. It shows an incremental change of the software failure rate after detected design fault elimination ($\lambda_{d(i)}$ vary by a constant $\Delta\lambda_{d(i)}$).
- All detected defects of software components are repaired immediately and no new defects are introduced. The mean time between failures and mean time to repair are exponentially distributed [15].
- Software testing datasets are updated after each test. The testing is performed on the complete body of input data.

The number of fragments *N* in the MFM depends on the number of expected undetected software faults n_i in *i* different software versions (1).

$$N_{fr} = \prod_{i=1}^{m}(n_i + 1), \quad (1)$$

The multi-fragmentation approach is described below.

1) Calculating the software failure $\lambda_{d(i)}$ and recovery $\mu_{d(i)}$ rates and evaluating the quantity of expected undetected software faults n_i in *i* different software versions.

2) Calculating the fragments number N_{fr} using (1).

3) Studying the system behavior, defining physical failure and recovery events and constructing system states and transitions based on those events.

4) Defining design failure and recovery events and constructing the system states and transitions based on those events.

5) Building the initial fragment (F_0) using the states and transitions developed in step *3*. The F_0 is a MC that illustrates the system operation process taking physical faults into account. It should be noted that every fragment also implicitly describes the amount of undetected software faults n_i in *i* different software versions, thus F_1 can be defined by the following set:

$$F_0 = \{(n_1, n_2, ..., n_m) \mid n_1 = i, n_2 = j, ..., n_3 = k, \; i, j, k \in N\}, \quad (2)$$

where $(n_1, n_2, ..., n_m)$, $n_1 = i$, $n_2 = j$, ..., $n_m = k$ is the amount $(i, j, ..., k)$ of undetected software faults in *m* different software versions.

6) Generating all the fragments using (3). Each fragment is defined similarly to the F_1 (2):

$$F = \{F_0, F_1, ..., F_n \mid n \in (\overline{0, N_{fr}}), n \in N\} \quad (3)$$

7) Transitions between the fragments occur based on the design failure and recovery events (step 4). In case of a fault in n_i software version the system proceeds from initial fragment F_0 to one of the states in the set M_i - the set of software failure states, and recovers to the fragment F_i. According to the taken assumptions, after each design failure elimination the $\lambda_{d(i)}$ is changed by a constant $\Delta\lambda_{d(i)}$ [14]. Thus, the fragment F_i contains corresponds to a reduced value of $\lambda_{d(i)}$.

The Fig. 1 shows the general multi-fragmental Markov (MFM) model built using this approach.

[1] IEC 61508 – Part 6 states on p. 58: "Efficient algorithms have been developed and implemented in software packages a long time ago in order to solve above equations. Then, when using this approach, the analyst can focus only on the building of the models and not on the underlying mathematics..."

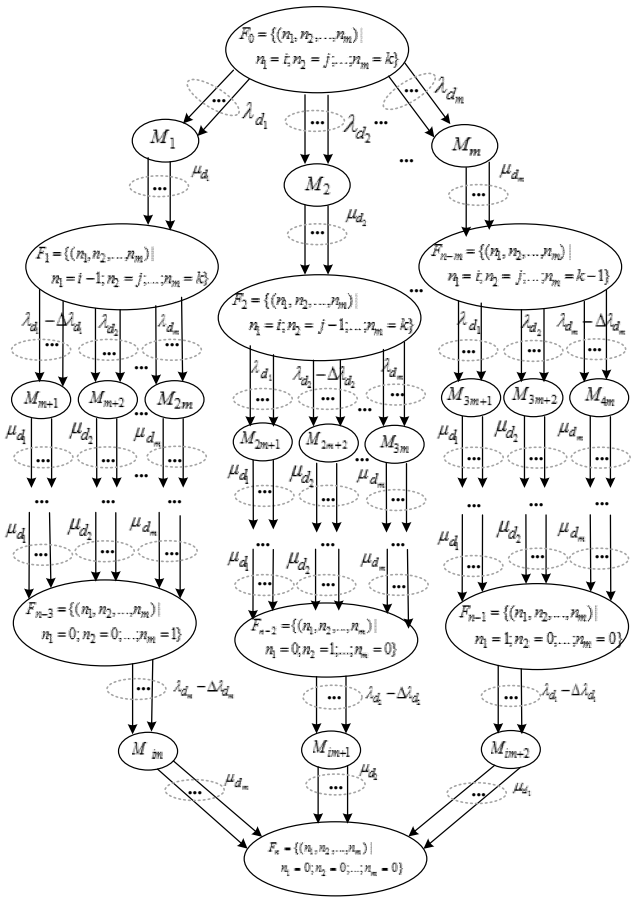

Fig. 1. General multi-fragmental Markov model

III. RESEARCH ARCHITEXTURE DESCRIPTION

Technology FPGA (Field Programmable Gates Arrays) is one of the most intensively developed and applied in I&Cs of nuclear power plants (NPPs) and other safety & mission critical domains [3].

The FPGA-based track is a basic component of both RTS architectures in the scope of this piece. Each track can contain up to 7 module types: analog and digital input modules (AIM, DIM); analog and digital output modules (AOM, DOM); logic module (LM); optical communication module (OCM); and analog input for neutron flux measurement module (AIFM). All modules are based on FPGA chips. The modules can be placed in 16 different positions on the track (two reserved positions for LM), using LVDS and fiber optical lines for internal/external communications. Such flexible redundancy management helps to ensure the high availability of the system. In this paper, we consider the tracks consisting of five modules: LM, DIM, DOM, AIM and AOM.

The Fig. 2 presents the structure diagram of a typical track. It is assumed that the corresponding components of all the tracks in the channels are identical, i.e. DIM on the 1st track is identical to the same module on other tracks in the channels, etc. The failure of the LM leads to the failure of the whole track, and failures of the DIM, DOM, AIM, AOM result in track malfunction. Therefore, it was assumed that failure of any module implies the general failed state of the track.

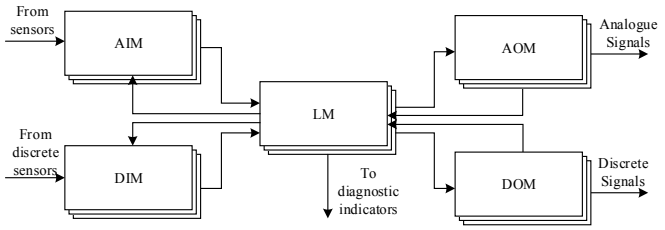

Fig. 2. The structure diagram of a typical track

The RDB for the two-channel three-track architecture is presented on Fig. 3. Fig. 4 presents the RBD for the three-channel two-track architecture. In both architectures, all tracks in the channels have identical hardware structures, but the software run on the system channels is diverse [10], i.e. non-identical but functionally equivalent software copies are deployed on the system channels. Each channel independently receives information from sensors and other NPP systems. The channels, each being capable of forming a reactor trip signal, are independent.

Reliability index $P_{pfi,j}$ determines hardware reliability of the track $T_{i,j}$ (defined by physical faults), where i indicates main ($T_{1,j}$) or diverse ($T_{2,j}$) channels, and j indicates the track number. Reliability index P_{dfi} determines software reliability of the main or diverse channels (defined by software faults), where i indicates the channel. Reliability index P_{mi}, determines reliability of the majority element m_i, where $i \in \overline{(1,4)}$.

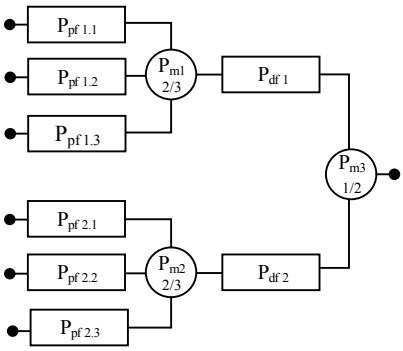

Fig. 3. Reliability-block diagram of two-channel tree-track RTS

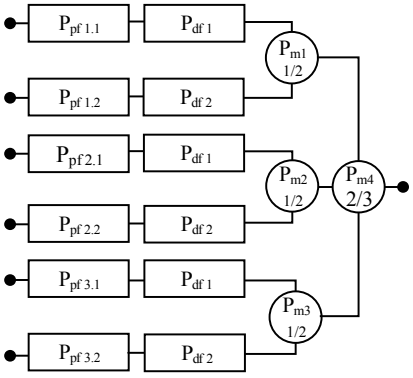

Fig. 4. Reliability-block diagram of three-channel two-track RTS

IV. RTS'S MARKOV MODELS. PHYSICAL FAULTS

This section presents the Markov models (MM) for the considered architectures taking physical faults into account. Fig. 5 presents the MM for the two-channel three-track RTS, and Fig. 6 – for the three-channel two-track RTS.

The following assumptions were used during the MMs development: each element of the research system at an arbitrary moment of time can only be in one of the two states – "working" and "failure"; the systems majority and control elements provide non-stop correct functioning.

Both models use the following parameters: $\lambda_{p(i,j)}$, $\mu_{p(i,j)}$, where $i \le 3$ and $j \le 3$ - the failure and repair rates for the failures caused by physical faults in the track $T_{i,j}$. As each track consists of five module types, the $\lambda_{p(i,j)}$ and $\mu_{p(i,j)}$ of the track $T_{i,j}$ can be calculated using (1) and (2) respectively:

$$\lambda_p(i,j) = \lambda_{DIM(i,j)} + \lambda_{DOM(i,j)} + \lambda_{LM(i,j)} + \lambda_{AIM(i,j)} + \lambda_{AOM(i,j)} \quad (4)$$

$$\mu_p(i,j) = \lambda_p(i,j)/(\lambda_{DIM(i,j)}/\mu_{DIM(i,j)} + \lambda_{DOM(i,j)}/\mu_{DOM(i,j)} + \lambda_{LM(i,j)}/\mu_{LM(i,j)} + \lambda_{AIM(i,j)}/\mu_{AIM(i,j)} + \lambda_{AOM(i,j)}/\mu_{AOM(i,j)}), \quad (5)$$

where $\{\lambda_{DIM(i,j)}, \lambda_{DOM(i,j)}, \lambda_{LM(i,j)}, \lambda_{AIM(i,j)}, \lambda_{AOM(i,j)}\}$ and $\{\mu_{DIM(i,j)}, \mu_{DOM(i,j)}, \mu_{LM(i,j)}, \mu_{AIM(i,j)}, \mu_{AOM(i,j)}\}$ are failure and repair rates caused by physical faults of DIM, DOM, LM, AIM, AOM, respectively. As all corresponding components of the tracks are identical, their failure and repair rates for the failures caused by physical faults are also equal. Thus, values of $\lambda_{p(i,j)}$, $\mu_{p(i,j)}$ are equal for all $T_{i,j}$ tracks.

A. Two-channel Three-track RTS

Observing the two-channel three-track RTS, we define the following set S_p ={(3; 3), (3; 2), (2; 3), (2; 2), (3; 1), (1; 3), (2; 1), (1; 2), (1; 1)}. Each member of this set presents the quantity of remaining working tracks in the main and diverse channels.

The system operation process is as follows. At time t_0 all tracks in both channels operate correctly in S_1. At a random moment t_n, a failure is caused by a physical fault in the track $T_{i,j}$ (for instance, in the main channel) and is subsequently detected by a majority component. The system moves to the S_2 state with a rate of $6\lambda_p$ and repairs back to the state S_1 with a rate of μ_p. The system functioning after state S_2 can unfold in two possible ways.

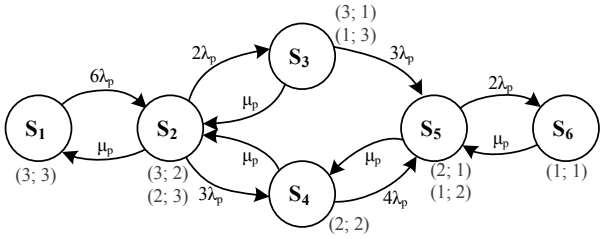

Fig. 5. Markov model of two-channel tree-track Reactor Trip System

- If during the $T_{i,j}$ track repair one of the remaining two tracks fails in the same channel (main channel), the system moves to the state S_3 with a rate of $2\lambda_p$ and recovers back to the state S_2 with a rate of μ_p. System moves from state S_3 to state S_5 if during the repair of two tracks in one (main) channel, a failure occurs in one of the tracks of another (diverse) channel.
- System moves from S_2 to the S_4 if during the $T_{i,j}$ track repair one of the tracks fails in another channel (diverse); the system moves into the state with a rate of $3\lambda_p$ and recovers back to the state with a rate of μ_p. If at the time of the last failed track repair, any new track in any channel fails, the system moves from state S_4 to state S_5 with a rate of $4\lambda_p$ and repairs back to S_4 with a rate of μ_p. It should be noted that the priority recovering strategy is repairing back to two working channels.

System moves from state S_5 to S_6 with a rate of $2\lambda_p$ and recovers back to S_5 with a rate of μ_p the last track fails in the remaining functioning channel. The MM consist of following states: $SF_w = \{\overline{S_1, S_5}\}$ – system working states; $SF_f=\{S_6\}$ – system failure state.

B. Single-channel Three-track RTS

For the three-channel two-track RTS, we define the following set S_p = {(2; 2; 2), (1; 2; 2), (2; 1; 2), (2; 2; 1), …, (0; 0; 1), (0; 1; 0), (0; 0; 1)}, members of present the quantity of working tracks in each channel.

The system operation process is as follows. At t_0 all tracks in the three channels operate correctly and deliver expected level of service (state S_1). At a random moment t_n a failure occurs due to a physical fault, and the system moves to the state S_2 with a failure rate of $6\lambda_p$ and repairs back to the state S_1 with a rate of μ_p. After the state S_2 the model presents two different cases of system operation process.

- If during the $T_{i,j}$ track repair the remaining track in the same channel fails, the system moves to the state S_4 with a rate of λ_p and recovers back to the state S_2 with a rate of μ_p. The system moves from S_4 to S_6 with a rate of $4\lambda_p$ if the physical failure occurs in one of the remaining two channels. If the same channel manifests yet another failure, the system moves from state S_6 to S_8 with a rate of λ_p, where S_8 is a non-working state of the system.
- If during the $T_{i,j}$ track repair the failure occurs in one of the remaining channels excluding already failed tracks, the system moves from S_2 to S_3 with a rate of $4\lambda_p$ and repairs back with a rate of μ_p. If during the last repair (state S_3) one more channel manifests a failure, the system proceeds from S_3 to S_5, representing the case when only one track still works in every channel,. The system transitions from S_5 to S_7 with a rate of $3\lambda_p$ if the last working track fails in one of the channels, and transitions to state S_9 if the failure occurs in yet another channel. The state S_9 is a nonworking state. We use the same recovering strategy as in the previous model (Fig. 4), which is represented by transitions between states S_6 and S_3 (Fig. 5).

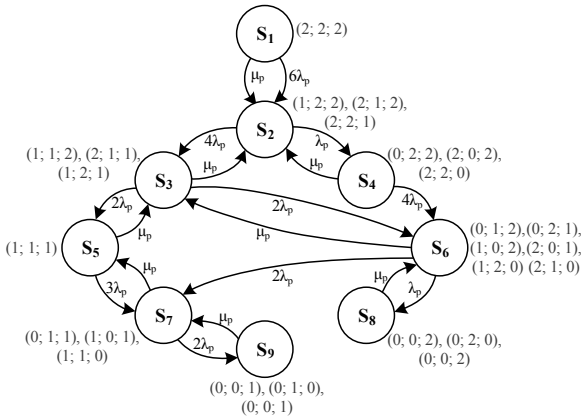

Fig. 6. Markov model of three-channel two-track Reactor Trip System

V. RTS'S MARKOV MODELS. DESIGN FAULTS

The Fig. 7 presents the uniform MFM of both RTS architectures taking into account design faults in diverse software versions.

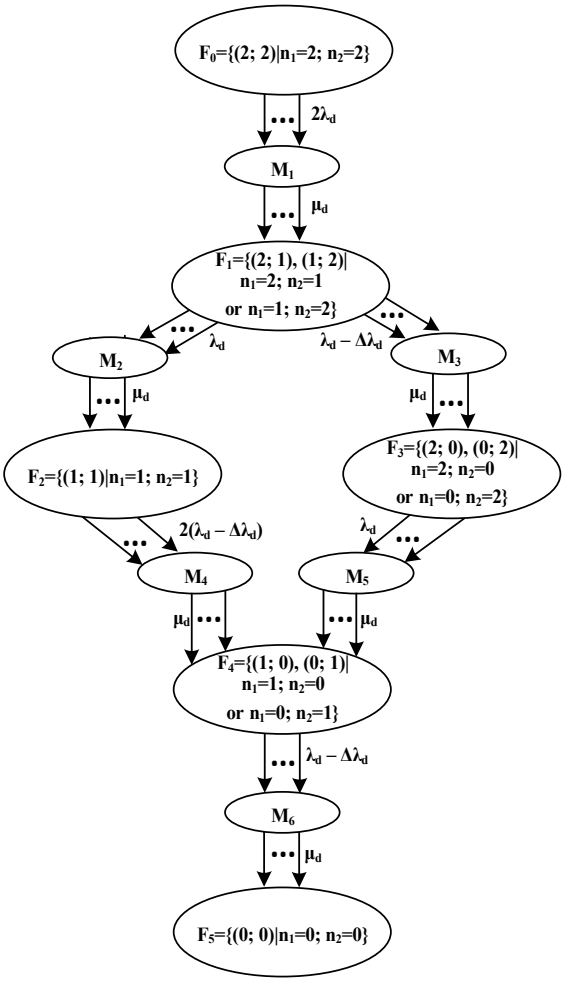

Fig. 7 Uniform multi-fragmental model for Reactor Trip System architectures

The main difference between those architectures is the type of F_0 fragment. The MM presented in Fig. 5 is used as F_0 for the two-channel three-track RTS, and MM in Fig. 6 – for the three-channel two-track RTS. Thus, the detailed MFM model for the first RTS architecture (two-channel three-track) is presented on the Fig. 8 and detailed MFM for three-track two-channel architecture on Fig.9. The system failed states are marked with grey color.

The following assumptions were used to build the MFM.

- Not more than two undetected software design faults are expected in the resulting software.
- The failure and repair rates for the failures caused by software design faults are equal (6-7). The steps of the failure rate decrease (after the channel recovery) are equal for both channels $\Delta\lambda_{d1} = \Delta\lambda_{d2}$ [15], [16]:

$$\lambda_{d1} = \lambda_{d2} \Rightarrow \lambda_d = \lambda_{d1} + \lambda_{d2}, \tag{6}$$

$$\mu_{d1} = \mu_{d2}; \mu_d = \lambda_d / (\sum_{i=1}^{2} \frac{\lambda_{d(i)}}{\mu_{d(i)}}) \tag{7}$$

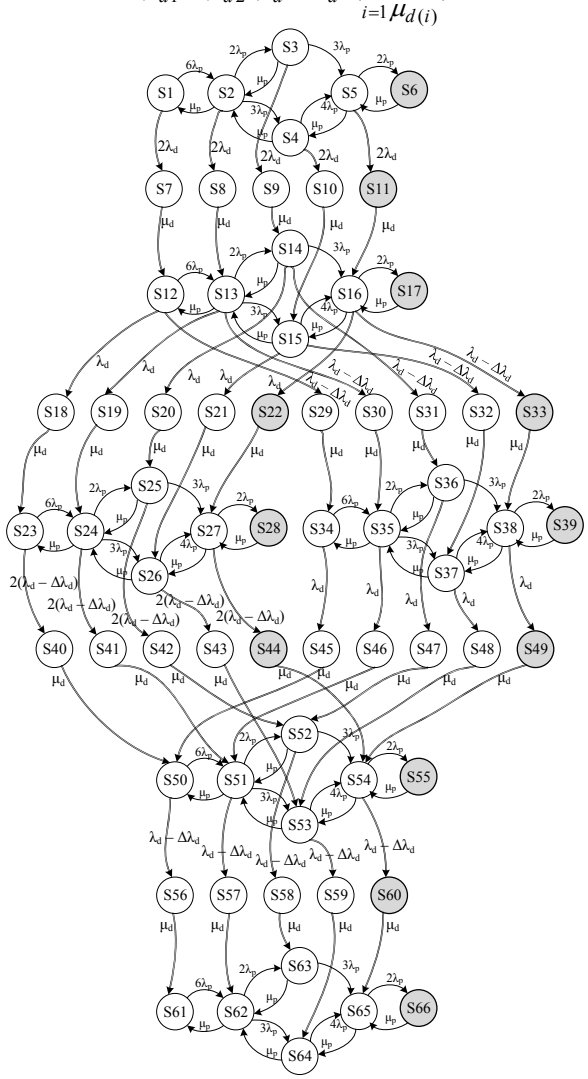

Fig. 8 The multi-fragmental model for two-channel three-track RTS architecture

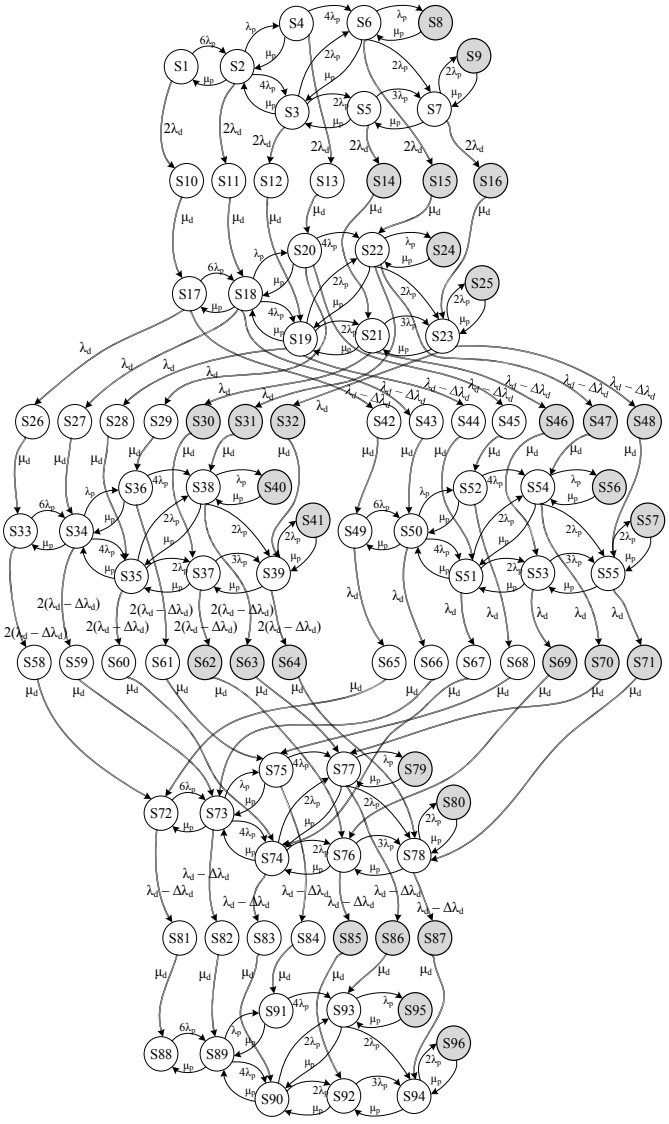

Fig. 9 The multi-fragmental model for three-channel two-track RTS architecture

VI. MODELS SOLUTION

One of the main computational difficulties affecting the use of numerical methods is a model's *stiffness*. There is no commonly adopted definition of "stiffness" but a few of the most widely used ones are summarized below.

The Cauchy problem $\frac{du}{dx} = F(x,u)$ is said to be stiff on the interval *[x0,X],* if there exists an *x* from this interval for which the following condition holds:

$$s(x) = \frac{\max\limits_{i=1,n} |\,\mathrm{Re}(\lambda_i)\,|}{\min\limits_{i=1,n} |\,\mathrm{Re}(\lambda_i)\,|} >> 1\,, \qquad (8)$$

where *s(x)* denotes the stiffness index and λ_i are the eigenvalues of the Jacobian matrix ($\mathrm{Re}\,\lambda_i < 0, i = 1,2,...,n$)[9].

The stiffness index of an MC was also defined in [6], [7] as the product of the *largest total exit rate* from a state and the *length of solution interval (=λ_it),* where λ_i are the eigenvalues of the Jacobian matrix. A system of differential equations (DE) is said to be stiff on the interval *[0,t)* if there exists a solution component of the system that has variation on that interval that is large enough compared to *1/t.* Thus, the length of the solution interval also becomes a measure of stiffness [7].

Many approaches have been developed to deal efficiently with MC stiffness [6, 8, 9]. They can be split into two groups - "stiffness-tolerance" (STA) and "stiffness-avoidance" approaches (SAA) [8]. Within STA, *specialized numerical methods* are used to provide highly accurate results in spite of stiffness. The SAA solution, on the other hand, is based on an *approximation algorithm,* which initially converts a stiff MC to a non-stiff chain that typically has a significantly smaller state space [8]. An advantage of this approach is that it can deal effectively with large stiff MCs. Achieving high accuracy, however, may be problematic with SAA,

In this section, we provide the results of MFMs (Fig.8 and Fig.9) solution using two approaches – stiffness-avoidance and stiffness-tolerance.

The MFMs (Fig. 8 and Fig. 9) were solved for the values:

- $\lambda_p = 10^{-4}$, $\mu_p = 1$, $\lambda_d = 5 \cdot 10^{-5}$, $\mu_d = 0.01$, $\Delta\lambda_d = 2.5 \cdot 10^{-5}$ and an accuracy requirement of 10^{-6}.
- Both models are of *moderate-stiffness* [10], with $s(x) = 1.667 \cdot 10^{-3}$ (8), with $max\ |Re(\lambda_i)| = 1.00063$ and $min\ |Re(\lambda_i)| = 0.0006$.

The assessment of availability function *A(t)* using STA for both MFMs was performed in mathematical packages Mathematica and Matlab using a built-in function implementing the implicit Runge-Kutta method. The solution was computed on the time interval of [0; 10 000] hours with a time step of *h=200* hours.

The MFMs were also solved using the SAA approach, essentially an aggregation/disaggregation technique developed by K. S. Trivedi, A. Bobbio and A. Reibmann [8].

The comparison of the results obtained by STA and SAA approaches for both RTS architectures are presented on Fig. 10.

Table 1 presents the average difference |ω| between STA and SAA *A(t)* results for both RTS architectures under study.

TABLE I. AVERAGE DIFFERENCE BETWEEN STA AND SAA RESULTS FOR BOTH RTS ARCHITECTURES

Approaches	RTS architectures / Average |ω|	
	Two-channel three-track	***Three-channel two-track***
Matlab – Mathematica	0,0001035	0,0003035
Matlab – SAA	0,0004001	0,0005417
Mathematica – SAA	0,0004002	0,0007179

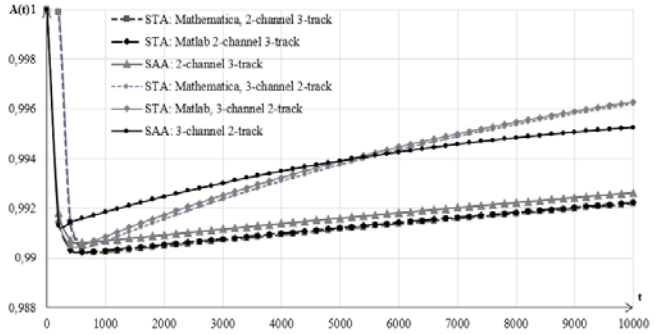

Fig. 10. Comparison of the A(t) results obtained using stiffness-avoidance and stiffness-tolerance approaches for three-channel RTS architecture

VII. CONCLUSIONS

We have discussed two models of the RTS architectures: the diverse two-channel three-track system and diverse three-channel two-track system. The models were built using multi-fragmental approach, which helps to illustrate the variation of system parameters. Based on the assessment results we can conclude that three-channel two-track diverse RTS provides more reliable functioning during the studied time period. The average difference between STA results showed the importance of its verification by the means of SAA technique. It should also be noted that in the process of selection from several math packages the usability becomes an important concern.

REFERENCE

[1] L. Pullum, "Software Fault Tolerance Techniques and Implementation", Artech House Computing Library, 2001.

[2] R. Wood, R. Belles, M Cetiner, et al, "Diversity Strategies for NPP I&C Systems", NUREG/CR-7007 ORNL/TM-2009/302, 2009.

[3] V. Kharchenko, A. Siora, V. Sklyar et al, "Multi-Diversity Versus Common Cause Failures "FPGA-Based Multi-Version NPP I&C Systems", Proc. of the 76th Conf. NPIC&HMIT, Las-Vegas, Nevada, USA, 2010.

[4] K. S. Trivedi, G. Ciardo, B. Dasarathy, M. Grottke, R. Matias, A. Rindos, B. Varshaw, "Achieving and Assuring High Availability". IEEE International symposium, IPDPS 2008, pp. 1-7.

[5] S. Archana, R. Srinivasan, K. S. Trivedi, "Availability Models in Practice", Proc. Int. Workshop on Fault-Tolerant Control and Computing (FTCC-1), May 22-23, Seoul, Korea, 2000.

[6] M. Malhotra, J. K. Muppala, K. S. Trivedi, "Stiffness-Tolerant Methods for Transient Analysis of Stiff Markov Chains", Microelectronic Reliability, vol.34(11), 1994, pp.1825-1841.

[7] A. Reibman, K. S. Trivedi, "Numerical Transient Analysis of Markov models", Comput. Opns. Res. vol.15(1), 1988, pp. 19-36.

[8] A. Bobbio, K. S. Trivedi, "A Aggregation Technique for Transient Analysis of Stiff Markov Chains", IEEE Trans. on Comp., C-35, 1986, pp. 803-814.

[9] O. Arushanyan, S. Zaletkin, "Numerical Solution of Ordinary Differential Equations using FORTRAN", Moscow State University, Moscow, 1990, p. 336.

[10] V. Kharchenko, O. Odarushchenko, V. Odarushchenko, P. Popov, "Availability Assessment of Computer Systems Described by Stiff Markov Chains: Case Study", Springer Verlang, Berlin-Heidelberg, CCIS , vol. 412, 2013, pp. 112 - 135 .

[11] K. S. Trivedi, R. Sahner, "SHARPE at the Age of Twenty Two", ACM SIGMETRICS Performance Evaluation Review, vol.36(4), 2009, pp. 52-57.

[12] E. Hairer, G. Wanner, "Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems", (Bank, R., Graham, R. I., Stoer, J., Varga, R., Yserentant, H. eds.), Springer-Verlang, 2010, p. 631.

[13] IEC 61508 (6 part), Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-Related Systems, 2010.

[14] Z. Jelinski, P. L. Moranda, "Software Reliability Research. Statistical Computer Performance Evaluation", W. Freiberger, ed., Academic press, New York, 1972, pp. 365 – 484.

[15] J. D. Musa, "Software Reliability. Measurement. Prediction. Application", McGraw-Hill Company, 1987, p. 395.

[16] V. Kharchenko, O. Odarushchenko, Y. Ponochovny, E. Odarushchenko, O. Kharibin, V. Odarushchenko, "High Availability Systems and Technologies" (Kharchenko, V. ed.), Lectures, National aerospace university "KhAI" Press, 2012, p. 249.

[17] P. Ganesh, "A Survey of Software Reliability Models", arXiv Preprint, 2002, p. 12.

[18] M. R. Lyu, "Handbook of Software Reliability Engineering", IEEE Computer Society Press, 1996, p. 819.