Multi-fragmentation Markov Modeling of a Reactor Trip System

Vyacheslav Kharchenko
Centre of Safety Infrastructure Oriented Research and Analysis
Kharkiv, Ukraine, 61000
v.kharchenko@csis.org.ua

Valentina Butenko¹
National Aerospace University “KhAI”
Department of Computers Systems and Networks
Kharkiv, Ukraine, 61000
v.odarushchenko@csn.khai.edu
ASME Student Member

Oleg Odarushchenko
Research and Production Corporation Radiy
Kirovograd, Ukraine, 25009
odarushchenko@gmail.com

Vladimir Sklyar
Research and Production Corporation Radiy
Kirovograd, Ukraine, 25009
vvsklyar@mail.ru

ABSTRACT

Markov models (MM) are widely used in dependability assessment of complex safety-critical systems. The main computational difficulties while using MM are model size and stiffness. Selection of the solution approach (SA) and tool based on analysis of MM stiffness and complexity increases the assessment accuracy.

This paper presents the safety assessment of NPP I&Cs: a two-channel FPGA-based Reactor Trip System with three parallel tracks on “2-out-of-3” logic. MM was build using multi-fragmentation approach and solved with several SAs and tools. The results analysis shows few application problems: importance of usability-oriented tool selection; achieving an accurate result; support the results verification.
INTRODUCTION

Dependability assessment of complex computer systems is an essential part of the development process as it either allows for demonstrating that relevant regulations have been met (e.g. as in safety critical applications) and/or for making informed decisions about the risks due to automation (e.g. in applications when poor dependability may lead to huge financial losses).

Achieving these goals, however, requires an \textit{accurate} assessment. Assessment errors may lead to wrong or suboptimal decisions.

Dependability and safety of such complex systems as NPP I&Cs can be assessed through model-based evaluation [1]. Such model-based evaluation can be performed through discrete-event simulation (DES), analytic models or combining simulation and analytical approaches. The main advantage of DES is ability to consider detailed system behavior in the model, while the drawback is long execution time, when accurate solution is needed. Analytical models tend to be more abstract, but are easier to develop and faster to solve than a DES models. The main difficulty is a necessity to set additional assumptions to make the models tractable [1]. Analytical modeling techniques can be split into two groups: state space (Markov chains, Petri nets, etc.) and non-state space (RBD, FTA, etc.) techniques. Selection of the appropriate technique is provided based on measurements of interest, level of components detalization, etc. The state space models are always preferred to non-space model, because they can easily incorporate realistic system behavior such as imperfect fault coverage, multiple failure modes, hot-swap components [2]. The RTS presented in this paper contains such
hot-swap components thus the non-state space models cannot be used. The MCs are mainly used for description of the processes for system physical components operation, interactions and interdependences. The software part in such models is usually assumed to provide correct functioning, because of the strongly managed by requirements and recommendations of international standards (see standard IEC 60880-2006, IEC 61508 - 2010) validation and verification (V&V) procedures. There are a lot of verification techniques applied during V&V process based on the documentation analysis, problem review, static analysis of code, testing and other [3]. Adding the aspect of software behavior into the research system MC can help to track its complex hardware-software interconnection.

Evaluating the system transient measures, such as availability function, in some cases, can provide more useful information than steady-state measures. Use of MC apparatus for modeling the transient behavior of the complex system can lead to the number of description and computational difficulties.

It is highly important to provide the correct verbal system presentation and use realistic assumptions and parameters. In case of modeling the system software components by MC the modeler has to introduce special assumptions [4, 5, 6, 7], which must keep the required abstraction level.

One of the main computational difficulties is the size of the models (i.e. their largeness, structural complexity), which leads to problems in its construction, storage and solution. Adding the software component behavior into the Markov model (MM) increases its size rapidly. Because of models size the closed-form solutions of transient
measures become infeasible, in this case, modeler can rely on the numerical methods or imitation modeling. Modeling components interaction enlarge the state space significantly, and results in sparse matrices of DE coefficients. A transient solution methods, which does not preserve sparsity is unacceptable for most large problems [8].

The next complexity in solving large models that effect on numerical solution results is the model stiffness [9]. It is an undesirable property of many practical MCs as it poses difficulties in finding transient solutions. In practice, stiffness in models of complex computer systems is caused by: (a) in case of repairable systems the rates of failure and repair differ by several orders of magnitude [10]; (b) fault-tolerant computer systems (CS) use redundancy. The rates of simultaneous failure of redundant components are typically significantly lower than the rates of the individual components [10]; (c) in models of reliability of modular software the modules’ failure rates are significantly lower than the rates of passing the control from a module to a module [10].

Several techniques were developed to deal efficiently with such MC largeness and stiffness [9, 10, 11] and a variety of tools can be used to find the MC solution [6, 12, 13]. Accurate selection of the solution approach and tool, based on analysis of the MC stiffness and complexity, increases the level of confidence for the assessment results [5].

This paper presents the case study for typical NPP I&C system, the Reactor Trip System (RTS) produced by RPC Radiy. This is a two-channel system with three parallel tracks on voting logic “2-out-of-3” in each channel. The multi-fragmental [7] approach was used to analyze and describe the ways how the software design faults can be
introduced into the system MM. The availability function, which consider the RTS failure and repair processes, was used as the dependability and safety are assessment parameter. The state space models (i.e. MM, etc.) are recommended [14] for evaluation of availability function for systems, which are used in critical domains. Several solution techniques and tools were used to analyze and describe the main difficulties in case of solving the complex, stiff MM, and how they can be avoid. Analysis of case study results using different SPs allows to formulate few application problems: importance of usability-oriented SP selection in case of solving complex MM; achieving an accurate result for stiff MCs; support the results verification to ensure the needed level of confidence.

The paper consist of the following sections: Section 1 provides the detailed research system description; Section 2 describe the RTS system MM taking into account only physical component, and both physical and software components; in Section 3 several solution techniques and software packages are used to solve the RTS MM. and verify the achieved results.

1. RESEARCHED SYSTEM DESCRIPTION

This section presents the description of a studied NPP I&C system produced by RPC Radiy. This is Reactor Trip System (RTS) with two-channel, three-track architecture, on voting logic “2-out-of-3” for tracks in each channel and “1-out-of-2” between channels. The FPGA-based track is a basic component of observed RTS. Generally, each track can contain up to 7 module types: analogue and digital input modules (AIM, DIM); analogue and digital output modules (AOM, DOM); logic module (LM); optical
communication module (OCM); and analogue input for neutron flux measurement module (AIFM). 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. Each channel independently receives information from sensors and other NPP systems. The channels, each being capable of forming a reactor trip signal, are independent.

Each track, observed in this paper, consists of five modules: LM, DIM, DOM, AIM and AOM. The Figure 1 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. 1. The structure diagram of a typical track

The RTS reliability-block diagram is presented on Fig. 2. Reliability index $P_{pf,i,j}$ 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 number of the track. Reliability index $P_{df,i}$ determines software reliability of the main or diverse channels (defined by software faults), where $i$ indicates channel. Reliability index $P_{maji}$ determines reliability of the majority element $maj_i$, where $i \in \{1,3\}$. 

Fig. 2. Reliability-block diagram of two-channel three-track system
Informally, the system operates as follows. Initially, all components (tracks) are working correctly and deliver the service as expected. If one of the tracks in main channel has failed, by majority voting the failure of the failed track is detected and while the failed track is being repaired, the operation will continue with the second and third tracks in channel. If before the first track has been repaired another one fails, the majority voting component will detect the output disagreement and main channel will stop. The diverse channel operates identically to the main. The operation process continues until one of channels works correctly. Clearly, the reliability of the majority components affects significantly the system reliability. In the observed RTS architecture the failure of majority components ($P_{m1}$ or $P_{m2}$) leads to the failure of the main or diverse channels, and to the general RTS failure ($P_{m3}$).

2. MODELING

2.1. RTS Markov model. Physical Faults

Figure 3 presents the MM of RTS system, which takes into account the physical and majority faults. 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” or “failure”; the control element provide non-stop correct functioning.

The model parameters are as follows:

(a) $\lambda_{p(i,j)}, \gamma_{p(i,j)}, i \in (1,2), j \in (1,3)$ - the failure and repair rates for the failures caused by physical faults in the track $T_{ij}$.
As each track consists of five module types, the $\lambda_{p(i,j)}$ and $\gamma_{p(i,j)}$ of the track $T_{i,j}$ can be calculated using (Eq.(1)) and (Eq.(2)) respectively:

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

(1)

$$\gamma_{p(i,j)} = \gamma_{DIM(i,j)} + \gamma_{DOM(i,j)} + \gamma_{LM(i,j)} + \gamma_{AIM(i,j)} + \gamma_{AOM(i,j)}$$

(2)

where $\{\lambda_{DIM(i,j)}, \lambda_{DOM(i,j)}, \lambda_{LM(i,j)}, \lambda_{AIM(i,j)}, \lambda_{AOM(i,j)}\}$ and $\{\gamma_{DIM(i,j)}, \gamma_{DOM(i,j)}, \gamma_{LM(i,j)}, \gamma_{AIM(i,j)}, \gamma_{AOM(i,j)}\}$ 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 $\lambda_{p(i,j)}$, $\gamma_{p(i,j)}$ are equal for all $T_{i,j}$ tracks.

(b) $\lambda_{maj(i)}$, $\gamma_{maj(i)}$, $i \in \{1, 2\}$ - failure and repair rates caused by physical faults of $m_i$ majority component. It is assumed that their values are also equal.

Set $S_p = \{(3; 3), (3; 2), (2; 3), (2; 2), (3; 1), (1; 3), (2; 1), (1; 2), (1; 1)\}$ (Fig. 3) 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 $\gamma_p$. The system functioning after state $S_2$ can unfold in two possible ways.
(a) 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 $\gamma_p$. 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.

(b) 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 $\gamma_p$. 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 $\gamma_p$. 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 $\gamma_p$ the last track fails in the remaining functioning channel. Transitions between states: $S_1$ - $S_7$, $S_2$ - $S_8$, $S_3$ - $S_9$, $S_4$ - $S_{10}$ and $S_5$ - $S_{11}$ illustrates the cases if during the main and diverse channels functioning one of the majority component fails.

The MM consist of following states: $SF_w = \{S_1, S_2\}$ – system working states; $SF_f = \{S_6, S_{11}\}$ – system failure states.

Fig. 3. MM Of RTS in case of physical faults

2.1. RTS Markov model. Design Faults

Similarly to the hardware components, the fault may occur in the software part. The observed RTS is FPGA-based, thus investigated software faults are such kinds
of faults, which are typical for VHDL coding process that were not covered by V&V procedure. In case of developing MMs that take into account software design faults the following assumptions were used.

(a) The failure rate of the design faults $\lambda_{d(i)}$ is proportional to their residual amount $n_i$ in $i$ – different software versions [4]. This assumption uses an incremental change of the software failure rate after detected design fault elimination ($\lambda_{d(i)}$ vary on a constant $\Delta\lambda_{d(i)}$), thus the Jelinski-Moranda model can be used [4].

(b) All detected defects are eliminated instantaneously and no new defects are introduced. The mean time between failures and mean time to repair are exponentially distributed [5].

(c) Software testing datasets are updated after each test. The testing is performed on the complete body of input data.

In MMs the failure rate variation can be illustrated using multi-fragmentation approach [6]. Using this approach the model is presented as $N$ fragments that have the same structure but may differ in one or more parameter values [7]. The number of fragments $F$ in the MC depends on the number of expected undetected software faults $n_i$ in $i$ – different software versions (Eq. (3)).

$$N_{\text{frag}} = \prod_{i=1}^{m} (n_i + 1)$$

(3)

The multi-fragmentation approach is described below.

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

(2) Calculating the fragments number $N_{\text{frag}}$ using (Eq.(3)).
(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.

Fig. 4. General MFM model

(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:

\[
F_0 = \{ (n_1, n_2, \ldots, n_m) | n_1 = i, n_2 = j, \ldots, n_m = k, \}
\]

\( i, j, k \in N \) \hspace{1cm} (4)

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

(6) Generating all the fragments using (Eq.(5)). Each fragment is defined similarly to the \( F_2 \) (Eq.(4)):

\[
F = \{ F_0, F_1, \ldots, F_n | n \in (0, N_{\text{fragments}}), n \in N \}
\]

(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 $\Delta \lambda_{d(i)}$ [4]. Thus, the fragment $F_i$ contains corresponds to a reduced value of $\lambda_{d(i)}$.

Figure 4 shows the general multi-fragmental Markov (MFM) model build using this approach. Figure 2 shows that software run on the system channels is diverse [15], i.e. non-identical but functionally equivalent software copies are deployed on the system channels. It is was assumed, that not more than two undetected software design faults can be expected at the resulting software.

Figure 5 presents the MFM model of RTS system taking into account design faults. The initial amount of undetected design faults ($n_1$ and $n_2$) in the main and diverse channels is equal two. On a purpose of reducing the MFM size two next assumptions were used.

(a) The failure and repair rates for the failures caused by software design faults are equal (Eq. (6 – 7)). Step of the failure rate decrease after the channel recovers from failure for both channels are equal $\Delta \lambda_{d1} = \Delta \lambda_{d2}$ (Eq. (8)) [7]:

$$\lambda_{d1} = \lambda_{d2} \Rightarrow \tilde{\lambda}_d = \hat{\lambda}_d + \lambda_{d2}.$$

(b) The systems voting majority element provides unstoppable correct functioning.

The MFM consist of the following operating states: $SF_{33} = \{S_1, S_{12}, S_{28}, S_{34}, S_{50}, S_{61}\}$ – states in which all tracks in both channels operate correctly; $SF_{23/32} = \{S_2, S_{13}, S_{29}, S_{35}, S_{51}, S_{62}\}$ – five tracks in both channels operate correctly; $SF_{13/31} = \{S_3, S_{14}, S_{30}, S_{36}, S_{52}, S_{61}\}$ –
$S_{63}$ – states in which one channel has failed; $SF_{22} = \{S_4, S_{15}, S_{37}, S_{53}, S_{64}\}$ – in both channels one track has failed; $SF_{21/12} = \{S_5, S_{16}, S_{38}, S_{54}, S_{65}\}$ – three tracks in both channels operate correctly; $SF_p = \{S_6, S_{17}, S_{39}, S_{55}, S_{66}\}$ – system failed states caused by physical failures; $SF_d = \{S_7 - S_{11}, S_{18} - S_{27}, S_{40} - S_{49}, S_{56} - S_{60}\}$ – states in which the software design failure occurs. Thus, the system working states are: $SF_w = \{S_1 - S_5, S_7, S_8, S_{10}, S_{12} - S_{16}, S_{18}, S_{19}, S_{21}, S_{23}, S_{24}, S_{26}, S_{28} - S_{32}, S_{34} - S_{38}, S_{40}, S_{41}, S_{43}, S_{45}, S_{46}, S_{48}, S_{50} - S_{54}, S_{56}, S_{57}, S_{59}, S_{61} - S_{65}\}$.

The system operation process can be shown using the description provided in Section 2.1 for MM on Fig. 3. Now Fig. 3 presents the initial fragment $F_0$ with two undetected software design faults in both channels (Eq. (8)):

$$F_0 = \{(n_1; n_2) | n_1 = 2; n_2 = 2\} \quad (8)$$

In the initial time $t_0$ the system provide the correct operation on both channels (fragment $F_0$ state $S_1$). If in one of the channels, the failure caused by design fault occurs the system moves to the state $S_7$ with rate $2\lambda_d$ and repairs to the state $S_{12}$ with rate $\gamma_d$. State $S_{12}$ is a state of first internal fragment $F_1$ (Eq. (9)):

$$F_1 = \{(n_1; n_2) | \{n_1 = 1; n_2 = 2\} or \{n_1 = 2; n_2 = 1\}\} \quad (9)$$

Fig. 5. MM of RTS in case of design faults

If the software failure occurs in the state $S_2$, which describes the case of five correctly functioning tracks, the system proceed to the state $S_8$ with rate $2\lambda_d$ and repairs to the fragment $F_1$ with rate $\gamma_d$. State $S_3$ in the initial fragment shows the failure of one channel (main or diverse). If on the working channel the failure caused by design fault
occurs the system moves to the failed state $S_9$ with rate $\lambda_d$ and repairs to the fragment $F_1$ (state $S_{14}$) with rate $\gamma_d$. State $S_4$ describe the case if on each channel one track has failed.

Transition from the state $S_4$ to the state $S_{20}$ occurs if the failure caused by design fault, was detected in one of the channels. Similarly to the previous cases, the system repairs to the fragment $F_1$ (state $S_{15}$). State $S_5$ describe the case if one of the channels has failed and the other one operates only on two tracks. System moves from state $S_5$ to the state $S_{11}$ with rate $\lambda_d$ if on the operating channel the failure caused by design fault occurs; after its elimination system proceed to the state $S_{16}$ with rate $\gamma_d$. State $S_6$ shows the situation when both channels have failed due to the physical faults.

Similarly, to the previous description, the system moves throw the fragments $F_1 - F_5$ (Eq. (9 – 13)) by detecting and eliminating all unexpected design faults.

$$F_2 = \{(n_1;n_2) |\{n_1 = 1; n_2 = 1\}\}, \quad (10)$$

$$F_3 = \{(n_1;n_2) |\{n_1 = 0; n_2 = 2\},\{n_1 = 2; n_2 = 0\}\}, \quad (11)$$

$$F_4 = \{(n_1;n_2) |\{n_1 = 0; n_2 = 1\},\{n_1 = 1; n_2 = 0\}\}, \quad (12)$$

$$F_5 = \{(n_1;n_2) |\{n_1 = 0; n_2 = 0\}\}. \quad (13)$$

From MFM (Fig. 5) the following system of differential equations (DE) can be derived. Here the DE for initial $F_0$, first internal $F_1$ and final $F_5$ (Eq.(14 - 16)) fragments are provided.

DE for the initial fragment $F_0$: 

NERS-14-1037 Valentina Butenko
\[ dP_1 / dt = -(6\lambda_p + 2\lambda_d) P_1(t) + \gamma_p P_2(t); \]
\[ dP_2 / dt = -(5\lambda_p + 2\lambda_d + \gamma_p) P_2(t) + 6\lambda_p P_1(t) + \gamma_p P_3(t) + \gamma_p P_4(t); \]
\[ dP_3 / dt = -(3\lambda_p + 2\lambda_d + \gamma_p) P_3(t) + 2\lambda_p P_2(t); \]
\[ dP_4 / dt = -(4\lambda_p + 2\lambda_d + \gamma_p) P_4(t) + 3\lambda_p P_2(t) + \gamma_p P_5(t); \]
\[ dP_5 / dt = -(2\lambda_p + \lambda_d + \gamma_p) P_5(t) + 4\lambda_p P_3(t) + 3\lambda_p P_3(t) + \gamma_p P_6(t); \]
\[ dP_6 / dt = -\gamma_p P_6(t) + 2\lambda_p P_5(t); \]

DE for the first internal fragment \( F_1 \):

\[ dP_{12} / dt = -(2\lambda_d - \Delta\lambda_d + 6\lambda_p) P_{12}(t) + \gamma_d P_7(t) + \gamma_p P_{13}(t); \]
\[ dP_{13} / dt = -(2\lambda_d - \Delta\lambda_d + 5\lambda_p + \gamma_p) P_{13}(t) + 6\lambda_p P_{12}(t) + \gamma_p P_{14}(t) + \gamma_p P_{15}(t) + \gamma_d P_9(t); \]
\[ dP_{14} / dt = -(2\lambda_d - \Delta\lambda_d + \gamma_p + 3\lambda_p) P_{14}(t) + 2\lambda_p P_{13}(t) + \gamma_d P_9(t); \]
\[ dP_{15} / dt = -(2\lambda_d - \Delta\lambda_d + 4\lambda_p + \gamma_p) P_{15}(t) + 3\lambda_p P_{13}(t) + \gamma_p P_{16}(t); \]
\[ dP_{16} / dt = -(2\lambda_d - \Delta\lambda_d + 2\lambda_p + \gamma_p) P_{16}(t) + 4\lambda_p P_{15}(t) + 3\lambda_p P_{14}(t) + \gamma_p P_{17}(t); \]
\[ dP_{17} / dt = -\gamma_p P_{17}(t) + 2\lambda_p P_{17}(t) \]

DE for the final fragment \( F_2 \):

\[ dP_{31} / dt = -6\lambda_p P_{31}(t) + \gamma_d P_{56}(t) + \gamma_p P_{62}(t); \]
\[ dP_{62} / dt = -(5\lambda_p + \gamma_p) P_{62}(t) + \gamma_d P_{57}(t) + 6\lambda_p P_{31}(t) + \gamma_p P_{63}(t) + \gamma_p P_{64}(t); \]
\[ dP_{63} / dt = -(3\lambda_p + \gamma_p) P_{63}(t) + \gamma_d P_{58}(t) + 2\lambda_p P_{62}(t); \]
\[ dP_{64} / dt = -(4\lambda_p + \gamma_p) P_{64}(t) + \gamma_d P_{59}(t) + 3\lambda_p P_{62}(t) + \gamma_p P_{65}(t); \]
\[ dP_{65} / dt = -(2\lambda_p + \gamma_p) P_{65}(t) + \gamma_d P_{66}(t) + 3\lambda_p P_{63}(t) + \gamma_p P_{67}(t) + 4\lambda_p P_{59}(t); \]
\[ dP_{66} / dt = -\gamma_d P_{66}(t) + 2\lambda_p P_{65}(t); \]

With the following initial conditions:

\[ P_1(0) = 1, P_i(0) = 0, i \in 2, 3, \ldots, 61 \]

3. Model Solution

NERS-14-1037 Valentina Butenko
One of the most common computational difficulty is MM is the size of state space. The number of states in model grow exponentially with the number of components in the system under study. Numerous high-level formalisms (languages) have been created to fill the gap between MC specification and system design specification, such as stochastic Petri nets [16, 17], stochastic process algebras [18], etc. There are automated tools [12], called state-space generators that convert the high-level specification of a model into equivalent underlying MC [19].

The next computational difficulty, which affects the use of numerical methods, is a models 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 $[x_0,X]$, if there exists an $x$ from this interval for which the following condition holds:

$$s(x) = \frac{\max_{i=1,n} |\text{Re}(\lambda_i)|}{\min_{i=1,n} |\text{Re}(\lambda_i)|} \gg 1,$$

where the $s(x)$ – denotes the stiffness index and $\lambda_i$ are the eigenvalues of a Jacobian matrix ($\text{Re} \lambda_i < 0, i = 1,2,..n$)[11].

Also the index of stiffness of an MC was defined in [8], [9] as the product of the largest total exit rate from a state and the length of solution interval (=\(\lambda t\)), where $\lambda_i$ 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 compared to $1/t$. Thus, the length of the solution
interval also becomes a measure of stiffness [8]. The previous empirical work [6] shows that quantitative value of \( s(x) \) have an impact on accuracy of different numerical methods – the higher \( s(x) \) value the more strict requirements imposed on the stability of chosen numerical method. The \( s(x) \) values can be split into three groups: high \( s(x) \geq 10^4 \), moderate \( 10^2 < s(x) < 10^4 \) and low \( s(x) < 10^2 \) [6]. Stiffness forces very small integration time steps, which increases the total computation time by several orders of magnitude and may result the different types of errors, such as round-off or truncation error. If the model is large, such increase of the solution iterations forces to use the additional storage place and schemes.

In order to avoid that an inefficient non-stiff numerical method waste too much effort when encountering stiffness, it is important to detect the stiffness at the early solution stage [20].

Many approaches have been developed to deal efficiently with MC stiffness [9, 10, 11]. They can be split into two groups - “stiffness-tolerance” (STA) and “stiffness-avoidance” approaches (SAA) [9]. With STA specialised numerical methods are used which provide highly accurate results despite stiffness. The SAA solution, on the other hand, is based on an approximation algorithm which converts a stiff MC to a non-stiff chain first, which typically has a significantly smaller state space [10]. An advantage of this approach is that it can deal effectively with large stiff MCs, but gives the approximate results, thus achieving high accuracy with SAA may be problematic.

During last three decades, a number of software tools have been developed and applied to solving models of complex systems. In general, they can be split into...
three main groups: specialized tools, e.g. MCs or other modeling apparatus software packages, off-the-shelf mathematical packages and own (user or modeler) developed utilities (ODU). SHARP [21], λPredict [13], Möbius [12] and etc. can be referred to as specialised tools. A number of off-the-shelf mathematical software packages exist which can be used for numerical MMs solution, e.g. Maple (Maplesoft), Mathematica (Wolfram Research), MATLAB (Mathworks), etc.

Such variety of tools and approaches is extremely helpful in the process of system modeling but poses also a difficulty when it comes to choosing the most appropriate one for a specific assessment. As every tool is limited in its properties and area of use a careful selection of the tools used to solve accurately and efficiently large stiff MCs is needed. One of leading standards in safety area IEC 61508-2010 (6th part) [22] asserts that efficient algorithms for MCs solution were developed earlier and implemented into SP, so the modeler needs to focus only on building the model and not on the underlying mathematics. It should be noted that stiffness usually requires modeler to focus on several math detail to avoid the use of inefficient approaches and tools. The previous work shows that $s(x)$ value can depend on the approach and tool selection procedure [6].

This section provides the description and comparison of the MFM (Fig. 5) model solution using both STA and SAA approaches and a set of math packages. Analysis of case study results using different software packages allows to formulate few application problems: importance of usability-oriented selection of the math package in
case of solving complex MM; achieving an accurate result for stiff MCs; support the results verification to ensure the needed level of confidence.

3.1. Stiffness-tolerance Approach

The main idea of this approach is using methods that are stable for solving stiff models. These can be split broadly into two classes: “classical” numerical methods for solution of stiff DEs and “modified” numerical methods used for finding a solution in special cases.

(a) The classical (non-modified) numerical methods for solving stiff DEs use special single-step and multi-step integration methods. Examples of such methods are the implicit Runge-Kutta, the TR-BDF2, the Rosenbrock method, the exponential method, the implicit Gir method described in [9], [11] and [23]. The implicit Runge-Kutta, TR-BDF2 and Rosenbrock method are implemented by several mathematical off-the-shelf software packages and are usually considered the most accurate methods for solving stiff ODEs.

(b) An example of the modified numerical methods is the exponential modified method. The original algorithm was presented in [11] and is based on the evaluation of the matrix exponent. In [11] this method is recommended as one of the most effective algorithms for solving the class of ODE systems with a high value of the Lipchitz constant, and as a special part of a stiff ODE. As a modification part, an automated adaptive step of integration can be implemented [11]. As the method has a multi-step algorithm the given modification can increase the accuracy of the solution. The amount
of computations and the machine time needed for the solution of stiff DEs can be reduced, too [11].

The solution provided by using any numerical method is expected to be accurate. However, typically the result obtained with numerical method include errors coming from different sources, such as: problem statement error, truncation error, round-off error, initial error. Ideally, the control of each of these components of the computational error is recommended.

This paper presents the availability assessment of the specified RTS architecture based on the developed MFM.

The MFM (Fig. 5) was solved for the values of \( \lambda_p = 10^{-4}, \gamma_p = 1, \lambda_d = 5 \cdot 10^{-5}, \gamma_d = 0.01, \Delta \lambda_d = 2.5 \cdot 10^{-5} \) and an accuracy requirement of \( 10^{-6} \). The model is of moderate-stiffness [5], with \( s(x) = 1.667 \cdot 10^3 \) (Eq. (18)), with \( \max |Re(\lambda_i)| = 1.00063 \) and \( \min |Re(\lambda_i)| = 0.0006 \).

The assessment of the availability function \( P_A(t) \) was provided using Mathematica, Maple, Matlab and EXPMETH. In the math packages the special function, which implements the implicit Runge-Kutta method was used.

The solution was computed on the time interval \([0; 10 000]\) hours with time-step \( h = 50 \) hours. The Fig. 6 presents the \( P_A(t) \) with \( h = 200 \) to increase the results readability.

The availability function \( P_A(t) \) is defined as a sum of probabilities of system working states \( (SF_w) \) (Eq. (19)). Using the initial conditions (Eq. (17)) : \( P_A(0) = 1 \).

\[
P_A(t) = \sum_{i=1}^{45} P_i(t), i \in SF_w
\]

(19)
Fig. 6. Comparison of the results obtained with STA on the $t = [0; 10000]$.

The Table 1 presents the average difference $|\omega|$ between Matlab, Maple and Mathematica for the achieved $P_A(t)$ values.

Table 1. Average difference between packages for $P_A(t)$ value

The usability-oriented selection of the mathematical packages is important in case of solving large MM. The mathematical packages offer convenience, but at the same time significant effort is required to construct the necessary function in case of large models, which may introduce scope of human errors, e.g. while entering the required data. For instance, Maple and Mathematica packages for the MFM solution use a matrix of DE coefficients, system of DE in explicit form and functions with five and seven arguments, respectively. MATLAB uses the matrix of DE coefficients and a function with four arguments.

The average difference (Table 1) between achieved $P_A(t)$ values in different packages, while using the same numerical method and initial data, shows the necessity of additional results verification. The achieved results highlight the necessity of careful selection of the software tools based on such criteria’s as interface usability, results portability and presence of special functions, which deals with such MC features as stiffness etc.

3.2. Stiffness-avoidance Approach

The basic idea of this approach is a model transformation by identifying and eliminating the stiffness from the model, which would bring two benefits: i) a reduction of the largeness of the initial MC, and ii) efficiency in solving a non-stiff model using
standard numerical methods. The approach was named an aggregation/disaggregation technique for transient solution of stiff MCs. The technique, developed by K. S. Trivedi, A. Bobbio and A. Reibmann [10], can be applied to any MC with transition rates that can be grouped into two separate sets of values – the set of slow and the set of fast states [10].

Table 2. Comparison of STA and SAA results

While the transformation of the initial stiff MC brings benefits in terms of efficiency, it is still important to take a systematic study of the impact of the transformation (from a stiff to a non-stiff MC on the accuracy of the solution). In addition, since the method is not supported by standard off-the-shelf tools, the scope for human error in applying it is non-negligible.

The solution of MFM (Fig. 5) was also obtained using SAA (aggregation/disaggregation technique [10]). Table 2, presents the probabilities of first two states in initial (states $S_1, S_2$ in $F_0$), first internal ($S_{12}, S_{13}$ in $F_1$) and final ($S_{61}, S_{62}$ in $F_5$) fragments of the MFM that were obtained using SAA in comparison to the results obtained with STA. As an example of STA solution, the MATLAB results were used. The average difference $|\omega|$ between results of SAA and STA (MATLAB) is also presented in Table 2.

Results of the comparison (Table 2) shows that SAA approach can be used to verify the results of STA approach, basically, for the results of the mathematical packages.
The main features of the SAA (stiffness elimination and largeness reduction) enlarge its for studying the large MM on the long solution intervals, while the STA use becomes infeasible due to the matrix sparsity and errors accumulation.

4. CONCLUSIONS

This paper presents the case study for assessment the availability parameter of NPP I&Cs, in particular, RTS. The system model was built using the MC apparatus, taking into account the failures caused by physical and design faults.

The multi-fragmental approach was used to describe the hardware-software components interconnection in case of variation of the failure rate caused by design faults. Several techniques and mathematical packages were used to solve the resulting complex stiff MC and show the main computational difficulties of their application. For achieving the accurate assessment results it is important to detect the model stiffness as early as possible, to avoid that an inefficient non-stiff numerical method waste too much effort when encountering stiffness. Case study results analysis allows to conclude that usability-oriented selection of the math package and careful results verification helps to ensure the needed level of confidence.
NOMENCLATURE

\( F \) set of model fragments

\( F_0 \) initial fragment

\( M \) set of software failure states

\( n \) residual amount of software faults

\( N_{\text{frag}} \) number of fragments in MM

\( P_{\text{maj}} \) reliability index of the majority element

\( P_{\text{pf}} \) hardware reliability index of the track \( T \)

\( SF_f \) set of system failure states

\( SF_w \) set of system working states

\( T \) system track

\( t \) time

\( t_0 \) initial time

Greek letters

\( \gamma \) failure recovery rate

\( \gamma_d \) the failure recovery rate of the design faults

\( \gamma_{\text{maj}} \) failure repair rate caused by physical faults of majority element

\( \gamma_p \) failure repair rate for the failures caused by physical faults in the track \( T \)

\( \Delta \lambda_d \) constant of \( \lambda_d \) variation

\( \lambda \) failure rate

\( \lambda_d \) the failure rate of the design faults

\( \lambda_{\text{maj}} \) failure rate caused by physical faults of majority element
\( \lambda_p \) failure rate for the failures caused by physical faults in the track \( T \)

\( \omega \) average difference between results from mathematical packages

**Subscripts or Superscripts**

\( A \) availability

\( d \) design

\( df \) design fault

\( f \) failed

\( frag \) fragment

\( i \) counting number

\( j \) counting number

\( maj \) majority

\( max \) maximum

\( min \) minimum

\( p \) physical

\( pf \) physical failure

\( w \) working

**Abbreviations and acronyms widely used in the text and list of references**

AIFM Analogue Input Neutron Flux Measurement Module

AIM Analogue Input Module

AOM Analogue Output Module

CCIS Communications in Computer and Information Science

DE Differential Equations
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES</td>
<td>Discrete-event Simulation</td>
</tr>
<tr>
<td>DIM</td>
<td>Discrete Input Module</td>
</tr>
<tr>
<td>DOM</td>
<td>Discrete Output Module</td>
</tr>
<tr>
<td>ECSS</td>
<td>Europe Cooperation for Space Standardization</td>
</tr>
<tr>
<td>FIT</td>
<td>Fault Insertion Test</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-programmable Gate Array</td>
</tr>
<tr>
<td>FTA</td>
<td>Fault Trees Analysis</td>
</tr>
<tr>
<td>FTCC</td>
<td>Fault-tolerant Control and Computers</td>
</tr>
<tr>
<td>I&amp;Cs</td>
<td>Instrumentation and Control System</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines (USA)</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission (Switzerland)</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers (USA)</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual Property</td>
</tr>
<tr>
<td>KhAI</td>
<td>Kharkiv Aviation Institute (Ukraine)</td>
</tr>
<tr>
<td>LM</td>
<td>Logic Module</td>
</tr>
<tr>
<td>LNCS</td>
<td>Lecture Notes in Computer Science</td>
</tr>
<tr>
<td>LVDS</td>
<td>Low-voltage Differential Signaling</td>
</tr>
<tr>
<td>MC</td>
<td>Markov Chain</td>
</tr>
<tr>
<td>MFM</td>
<td>Multi-fragmental Markov Model</td>
</tr>
<tr>
<td>MM</td>
<td>Markov Model</td>
</tr>
<tr>
<td>NPP</td>
<td>Nuclear Power Plant</td>
</tr>
<tr>
<td>NUREG</td>
<td>Nuclear Regulatory Guide</td>
</tr>
</tbody>
</table>
OCM Optical Communication Module
ODE Ordinary Differential Equations
ODU Own Developed Utilities
RBD Reliability Block Diagram
Re real part
RPC Research and Production Corporation
RT&A Reliability Theory and Applications
RTS Reactor Trip Systems
SA Solution Approach
SAA Stiffness-avoidance Approach
SIL Safety Integrity Level
SP Software Package
STA Stiffness-tolerance Approach
TR-BDF2 Trapezoidal Rule with the Second Order Backward Difference Formula
US or USA United States of America
V&V Verification and Validation
VHDL Very High Speed Integrated Circuit Hardware Description Language
REFERENCES


Table Caption List

Table 1 Average difference between packages for $P_A(t)$ value
Table 2 Comparison of STA and SAA results
Figure Caption List

Fig. 1  The structure diagram of a typical track

Fig. 2  Reliability-block diagram of two-channel three-track system

Fig. 3  MM Of RTS in case of physical faults

Fig. 4  General MFM model

Fig. 5  MM of RTS in case of design faults

Fig. 6  Comparison of the results obtained with STA on the $t = [0; 10000]$
Table 1. Average difference between packages for $P_A(t)$ value

| Math-packages          | Average $|\omega|$ |
|-----------------------|------------|
| MATLAB-MAPLE          | 0,000014   |
| MATLAB-MATHEMATICA    | 0,000252   |
| MAPLE - MATHEMATICA   | 0,000251   |
Table 2. Comparison of STA and SAA results

<table>
<thead>
<tr>
<th>t</th>
<th>Approach</th>
<th>S1</th>
<th>S2</th>
<th>S12</th>
<th>S13</th>
<th>S61</th>
<th>S62</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>STA</td>
<td>0.979610</td>
<td>6.692·10^{-6}</td>
<td>0.000588</td>
<td>0.011153</td>
<td>3.4·10^{-12}</td>
<td>2.10^{-11}</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.979610</td>
<td>1.173·10^{-3}</td>
<td>0.000587</td>
<td>0.019543</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>STA</td>
<td>0.904294</td>
<td>4.871·10^{-3}</td>
<td>0.000543</td>
<td>0.081189</td>
<td>1.02·10^{-7}</td>
<td>6.1·10^{-11}</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.904294</td>
<td>5.358·10^{-3}</td>
<td>0.000542</td>
<td>0.089308</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2000</td>
<td>STA</td>
<td>0.818239</td>
<td>0.979610</td>
<td>1.173·10^{-5}</td>
<td>0.000587</td>
<td>0.153275</td>
<td>3.36·10^{-6}</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.818239</td>
<td>1.173·10^{-5}</td>
<td>0.000543</td>
<td>0.019543</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3000</td>
<td>STA</td>
<td>0.740373</td>
<td>0.904294</td>
<td>4.871·10^{-3}</td>
<td>0.000543</td>
<td>0.209123</td>
<td>2.11·10^{-5}</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.740373</td>
<td>5.358·10^{-3}</td>
<td>0.000542</td>
<td>0.089308</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4000</td>
<td>STA</td>
<td>0.669917</td>
<td>0.818239</td>
<td>1.173·10^{-5}</td>
<td>0.000587</td>
<td>0.251380</td>
<td>7.2·10^{-5}</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.669917</td>
<td>1.173·10^{-5}</td>
<td>0.000543</td>
<td>0.019543</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5000</td>
<td>STA</td>
<td>0.606166</td>
<td>0.740373</td>
<td>4.871·10^{-3}</td>
<td>0.000543</td>
<td>0.282312</td>
<td>0.000180</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.606166</td>
<td>5.358·10^{-3}</td>
<td>0.000542</td>
<td>0.089308</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6000</td>
<td>STA</td>
<td>0.548482</td>
<td>0.669917</td>
<td>1.173·10^{-5}</td>
<td>0.000587</td>
<td>0.284906</td>
<td>0.000243</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.548482</td>
<td>1.173·10^{-5}</td>
<td>0.000543</td>
<td>0.019543</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7000</td>
<td>STA</td>
<td>0.496287</td>
<td>0.606166</td>
<td>4.871·10^{-3}</td>
<td>0.000543</td>
<td>0.303856</td>
<td>0.000373</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.496287</td>
<td>5.358·10^{-3}</td>
<td>0.000542</td>
<td>0.089308</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8000</td>
<td>STA</td>
<td>0.449059</td>
<td>0.548482</td>
<td>1.173·10^{-5}</td>
<td>0.000587</td>
<td>0.318701</td>
<td>0.000835</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.449059</td>
<td>1.173·10^{-5}</td>
<td>0.000543</td>
<td>0.019543</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9000</td>
<td>STA</td>
<td>0.406325</td>
<td>0.496287</td>
<td>4.871·10^{-3}</td>
<td>0.000543</td>
<td>0.327463</td>
<td>0.001126</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.406325</td>
<td>5.358·10^{-3}</td>
<td>0.000542</td>
<td>0.089308</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10000</td>
<td>STA</td>
<td>0.367658</td>
<td>0.449059</td>
<td>1.173·10^{-5}</td>
<td>0.000587</td>
<td>0.325602</td>
<td>0.001348</td>
</tr>
<tr>
<td></td>
<td>SAA</td>
<td>0.367658</td>
<td>1.173·10^{-5}</td>
<td>0.000543</td>
<td>0.019543</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>1.82·10^{-7}</td>
<td>1.97·10^{-4}</td>
<td>6.36·10^{-5}</td>
<td>0.003541</td>
<td>0.001116</td>
<td>7.003·10^{-8}</td>
</tr>
</tbody>
</table>
Fig. 1. The structure diagram of a typical track
Fig. 2. Reliability-block diagram of two-channel three-track system
Fig. 3. MM of RTS in case of physical faults
Fig. 4. General MFM model
Fig. 5. MM of RTS in case of design faults
Fig. 6. Comparison of the results obtained with STA on the $t=[0; 10000]$