TSAR - Tool for System Availability and Reliability Analysis

Marko Lacković, Marije Ljolje, Robert Inkret

The article describes the TSAR tool for system availability and availability analysis. Failure and repair model is based on Markov chain defined by states and transitions. Each state can be interpreted as working or broken. Transitions are defined by transition probabilities and transition time probability density functions. Availability and reliability analysis is based on discrete event Monte Carlo simulation or Abraham's analytic algorithm. System state being a function of comprised module states serves as the base of simulation calculation, while the analytical calculation uses logical terms representing all paths between start and end modules. The tool is implemented in the *Cosmos* simulation kernel and *Cosmos* GUI.

***Index Terms*—analytic, availability, reliability, simulation**

I. Introduction

System availability and reliability are one of the primary parameters that should be taken into consideration when designing and implementing a telecommunication system. These systems have grown in complexity and price over the past decades following the growth of consumer needs and dramatic increase of exchanged data. Unreliable telecommunication system increases costs of the maintenance. Having in mind that the customer support is the most time and money consuming stage of system's life, it is reasonable for the manufacturer and/or service provider to put more effort in assuring that the system would work during it's expected lifetime.

The definition of a system can include hardware, software or a service. The failure of a system in all those cases remain virtually the same – it is a state when system doesn't provide satisfying service to its user. The satisfaction with the service can be determined by the goal function of the system, which can be defined according the system type. In the case of hardware it can be a function that produces output from the input, in the case of software it can be advertised function of a program, and in the case of service it can be a contract between a service user and a service provider. System availability and reliability includes all those cases and are strongly correlated to the quality of service which is of the primary interest to system's seller and system's buyer.

It is clear that the notion of system's failures and its influence on the system availability and reliability as important measure of the system quality has very important place in telecommunication curriculum. Our goal was to design and implement a simple, yet flexible model of a component behavior considering failures and repairs, and to analyze the behavior of a system that comprises such simple modules. This was supplemented by a graphical user interface that would allow simple system definition, parameterization of modules it contains and analysis of results.

The final tool was named TSAR – Tool for System Availability and Reliability analysis.

II. Availability and Reliability model

Availability is defined as the probability that a component/system will work in some point of time. Reliability is probability that a component/system will work properly during some defined period of time [1]. It is clear that the same model can be applied in both cases, but the calculations differ. The term to work properly will be shortened in the following text just to work, implying that the states where component/system works improperly will be considered as failures.

The model was implemented as a Markov chain, or a state and transition diagram with transition probabilities.

Figure 1 depicts a model comprising three states. Transitions between states are marked with (*xy*) where *x* denotes initial, and *y* denotes final state.

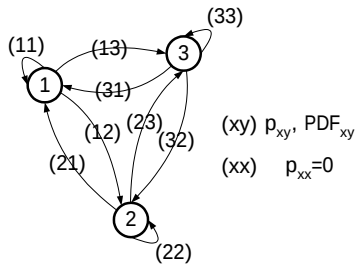

Figure 1 A three state model

Transition (*xy*) is determined by the transition probability and the time period a module stays in some state before making this particular transition. The sum of all transition probabilities from one state is equal to 1. Probabilities of transitions into the same state ((*xx*) transitions) are equal to 0 because they don't contribute to changes of module behavior and are equal to staying in the same state. Time that a module spends in a state depends on the transition that would change (exit from) that state. That time is determined by the probability density function (PDF) type (e.g. exponential, uniform) and a mean transition time value. The whole model

is defined with two matrices and one list – transition probability matrix *P* and a PDF matrix defining each transition (Figure 2), and a state definition list *S* defining the nature of each state (working or broken). Figure 2 depicts this parameterization for a general model with *N* states.

$$P=\begin{bmatrix} 0 & p_{12} & \dots & p_{1N} \\ p_{21} & 0 & \dots & p_{2N} \\ . & & \dots & . \\ . & & \dots & . \\ p_{N1} & p_{N2} & \dots & 0 \end{bmatrix} \quad PDF=\begin{bmatrix} 0 & PDF_{12} & \dots & PDF_{1N} \\ PDF_{21} & 0 & \dots & PDF_{2N} \\ . & & \dots & . \\ . & & \dots & . \\ PDF_{N1} & PDF_{N2} & \dots & 0 \end{bmatrix} \quad \sum_j p_{ij}=1$$

$$PDF_{xy}=\begin{cases} PDF\ type \\ PDF\ mean\ value \end{cases} \qquad S_i \in \{0,1\}$$

$$S=[S_1,S_2,\dots,S_N]$$

Figure 2 Module parameterization

Figure 3 shows a mechanism implemented in each module. When a module gets to a state (including initial) a transition to some other state is calculated. Egress transition PDF gives enough information to calculate the period of time a module is going to spend in the current state.

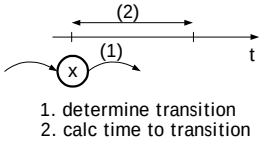

Figure 3 Transition mechanism

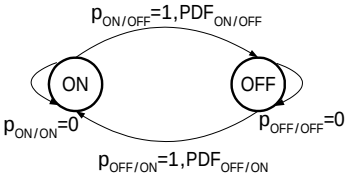

Figure 4 ON/OFF model

A. *ON/OFF Model*

ON/OFF model is a special case of the general model described in the last part. It is representative because failures are simulated in majority of cases with just two states (Figure 4). One of the states is defined as working state (*ON*) and the other as a failure state (*OFF*).

General model with *N* states includes several states denoted as working states. These states represent module wear-off over time. These states are omitted in the ON/OFF module and are aggregated in the ON state. This model represents a good starting point for explaining the concept of failures and repairs. Mean value for the $PDF_{ON/OFF}$ is denoted as the mean time to failure (*MTTF*), and mean value for the $PDF_{OFF/ON}$ as the mean time to repair (*MTTR*).

The ON/OFF model is used in the cases where detailed modeling isn't possible, like when modeling technical devices. This representation using states is a great simplification of a very complex device behavior influenced by a number of factors including external influences (heat, humidity) and stochastic physical processes in the material the device is made from. Their internal states aren't easily visible, and are usually simplified to the ON/OFF model. Time to failure and times to repair are easily measurable on the set of many identical components, and *MTTF* and *MTTR* are easily calculated from the measured data.

It is clear that the usage of models other than ON/OFF model is limited to the educational purposes, as the empirical data needed to construct such model for some real component is difficult to measure.

III. SYSTEM DESIGN AND ANALYIS

Figure 5 represents general system structure. *S* and *T* modules are one state modules that are always working properly and serve as edge points for system definition. All system modules have at least one direct or indirect connection to *S* and *T* module.

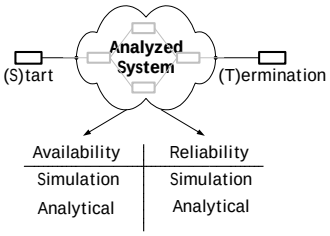

Figure 5 General system structure

Both availability (*A*) and reliability (*R*) analysis are supported on the defined system. They are both based on the notion of failure and repair of modules making the systems. That failure/repair model is generalized to the system level during the A/R analysis. The way of failure times measurement differs between A and R calculation.

A/R calculation can be carried out using the simulation or analytical methods.

A. *Simulation*

A Monte Carlo simulation has been used to make A/R calculations. Simulation is used in cases where analytical calculation would be too time consuming due to the great calculation complexity for large systems. Simulation introduces inaccuracy in the final results, which depends on the number of simulated failures in the system (simulation iterations).

System A/R calculation requires definition of the system state (working or broken). A system state was defined as the existence of at least one path between *S* and *T* modules. Modules in the system that are in the state declared as the failure state represent a break of path. This is an implicit definition of system states.

The simulation is based on detecting the system state changes which depends on changes of the modules' states. This is based on the discrete event simulation mechanism.

1) *Availability*

The availability calculation is based on calculating the total period the system was working (T_{ON}) as the sum of all working periods (Figure 6).

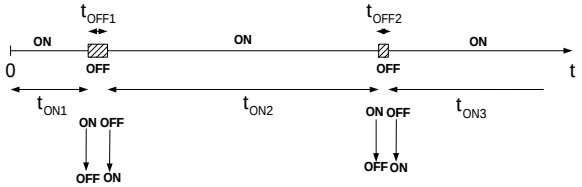

Figure 6 Availability simulation

The number of system state changes depend on the number of simulation iterations. Each iteration represents a change of state in one of the modules. That doesn't necessarily change the system state. The accuracy of the calculation depends on the number of iterations as the systems availability expression is equal to

$$A = \frac{T_{ON}}{total\ simulation\ time} = \frac{\sum_{i=0}^{\#changes} t_{ONi}}{total\ simulation\ time}. \quad (1)$$

2) Reliability

Reliability calculation depends on the time period the reliability is calculated for. The basis for calculation is the same as for the availability calculation. Once when system goes in the failure state the simulation is initialized and run one again (Figure 7). The simulation iteration here corresponds to the number of initializations.

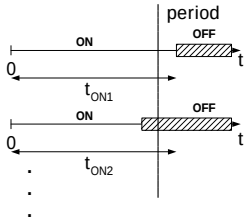

Figure 7 Reliability simulation

The system reliability is equal to the ratio of iterations that produced system working time longer than the specified period and the total number of iterations:

$$R(period) = \frac{\#(iterations\ with\ t_{ON} > period)}{\#iterations} \quad (2)$$

B. Analytical Calculation

The analytical calculation uses logical expressions to calculate the availability or reliability. Each expression represents one path between modules *S* and *T*.

Figure 8 depicts steps of the analytical calculation. The system is shown on the left side and comprises 6 modules (module *m*1 is connected to module *S* and module *m*6 to module *T*). There are two logical paths denoted as *P*1 and *P*2, which can be expressed as

$$P1 = V_{m1} \wedge V_{m2} \wedge V_{m3} \wedge V_{m5} \wedge V_{m6},$$
$$P2 = V_{m1} \wedge V_{m2} \wedge V_{m4} \wedge V_{m5} \wedge V_{m6}. \quad (3)$$

The V_{mx} stands for the module availability A_{mx} or module reliability R_{mx} depending on the type of calculation. System availability or reliability is equal to

$$V_s = P1 \vee P2 \quad (4)$$

Expressions in the sum aren't necessarily disjunctive. This fact makes the calculation difficult, and Abraham's algorithm [2] is used to represent the upper equation as the sum of disjunctive factors. This simplifies the calculation, but the algorithm is time consuming when it comes to a large number of factors in (3) and (4). This number depends on the number of modules in the system, and module connectivity (number of paths between *S* and *T*).

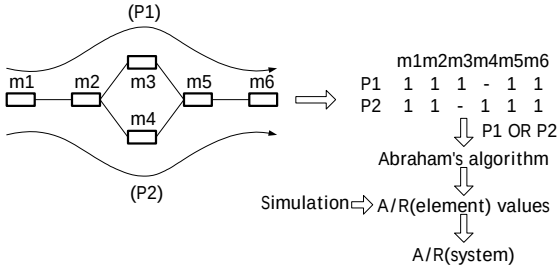

Figure 8 Steps of the analytical calculation procedure

The last step is to insert the availability/reliability values for each module. These can be explicitly given for each module, or can be calculated using described availability or reliability calculation for each module.

IV. TSAR STRUCTURE

TSAR was coded in C++ as an application using a *Cosmos* framework and simulation kernel [3]. Described models were implemented in two classes inheriting the `cModule` class [3]. `arModule` class implements the general model, and `arOnOffModule` class inherited from `arModule` class implements the ON/OFF model by restricting the general model to just two states (Figure 9). State and transition diagram was implemented using the `mcsStates` class [4], which contains two matrices defining transitions (probability and a PDF matrix), and a state description list. The system was implemented on the basis of `eventSystem` class [3], which incorporates the `eventDomain` class [4] necessary for the discrete event simulation.

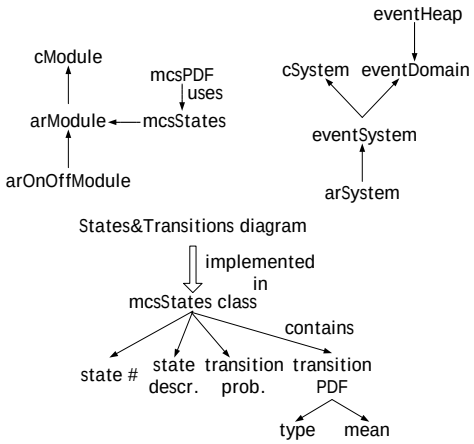

Figure 9 TSAR's tacsonomy

V. APPLICATION USAGE

Figure 10 shows the main window of the TSAR application. It is based on the general purpose Cosmos GUI and contains common toolbar, drawing area and output area. Module toolbar shown on the left side contains `arModule` and `arOnOffModule` classes as templates for `arSystem` structure definition.

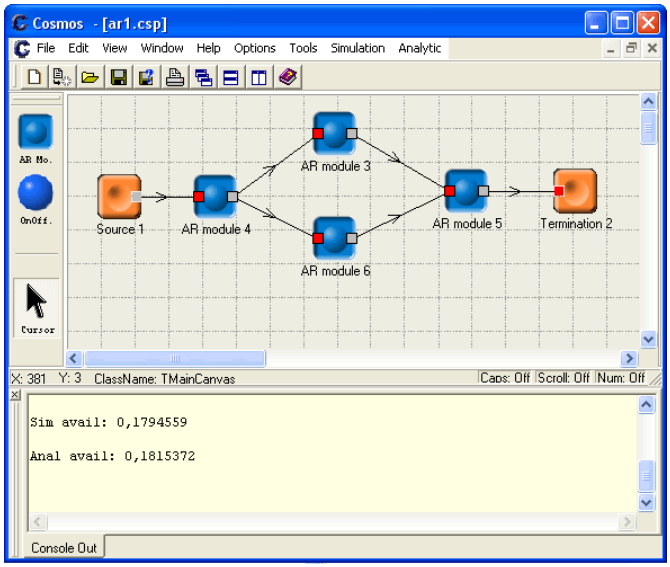

Figure 10 Application main window

A simple system structure with 4 `arModule` based components has been created (Figure 10). Borders of the system are connected to test nodes colored with different color (*S* and *T* modules).

Each module has its own property inspector containing all parameterized features (Figure 11). Simple type properties (e.g. scalar types) are shown directly in the inspector, while the complex properties (like matrices and PDFs) are shown in the form of the tree containing all their scalar attributes.

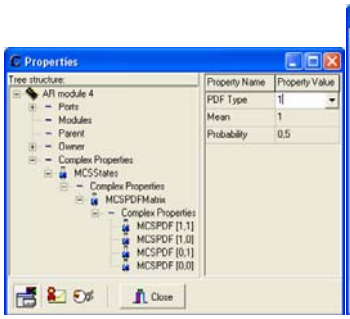

Figure 11 Module properties inspector

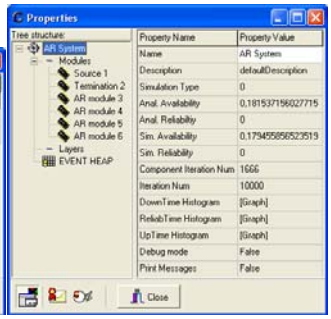

Figure 12 System properties dialog box

System property inspector if formed like module property inspector (Figure 12). It contains results for analytical/simulation availability/reliability along with list of all contained modules.

Analytical calculation can be conducted using explicitly determined module availability/reliability data, or by using simulation to determine availability/reliability for each module from Markov chain parameters (Figure 13).

Figure 14 shows the histogram of module down times. Each module contains histograms of up/down times in availability simulation, and reliability times in reliability simulation. These histograms are also present in the system.

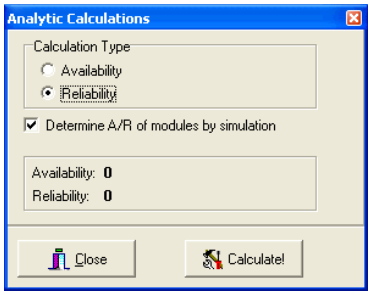

Figure 13 Analitical calculation main window

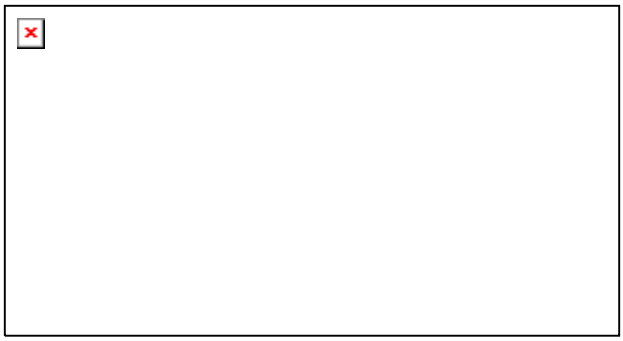

Figure 14 Module down times

VI. Conclusions and Future Work

This paper presented the main features of the TSAR tool. It is focused on system availability/reliability calculation using analytical procedure or discrete event Monte Carlo simulation. Analytical calculation is based on the Abraham's algorithm. Simulation model is based on the Markov chain model employing states and transitions. Each state can be defined as working or broken. Transitions are defined by the transition probability, and transition time determined by the probability density function. General Markov chain model is implemented in the `arModule` which can serve for complex failure and repair behavior. Model with on and off state is implemented in `arOnOffModule`.

Future work includes defining module behavior using a subgraph containing other modules (a network of modules). Introducing failure domains can describe failure dependencies. A failure of one module can cause a failure of another with some probability what is very important for telecommunication system design (like dependency of cable cuts and contained fiber failure).

References

[1] M.L. Schooman, *Probabilistic Reliability: An Engineering Approach,* 2nd ed., McGraw-Hill Book Company, 1968.

[2] J.A. Abraham. "An improved algorithm for network reliability", *IEEE Transactions on Reliability*, R-28:58-61, April 1979.

[3] M. Lackovic, R. Inkret, Network design, optimization and simulation tool *Cosmos*, *Proceedings of WAON*, June 13-14, 2001, Zagreb, Croatia, pp. 37-44.

[4] M. Lackovic, R. Inkret, B. Mikac, An object-oriented approach to telecommunication network modeling, *Proceedings of ESM*, June 3-5, 2002, Darmstadt, Germany