Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

J-SAEDES: A JAVA-BASED SIMULATION SOFTWARE TO IMPROVE RELIABILITY AND AVAILABILITY OF COMPUTER SYSTEMS AND NETWORKS

Angel A. Juan

Dept. Applied Mathematics - EPSEB
Technical Univ. of Catalonia
Av. Doctor Marañón, 44-50
Barcelona, 08028, SPAIN

Javier Faulin

Dept. Statistics and OR
Public University of Navarre
Campus Arrosadia
Pamplona, 31006, SPAIN

Joan M. Marquès

Dept. of Computer Science
Open University of Catalonia
Rambla Poblenou, 156
Barcelona, 08018, SPAIN

Mateo Sorroche

Dept. Electronic Engineering - EPSEM
Technical Univ. of Catalonia
Av. Bases de Manresa, 61-73
Manresa, 08242, SPAIN

ABSTRACT

Nowadays, many companies rely on computer systems and networks for functions such as order entry, customer support, supply chain management, and employee administration. Therefore, reliability and availability of such systems and networks are becoming critical factors for the present-day enterprise or institution. Nevertheless, the task of determining reliability/availability levels for time-dependent systems and networks at the design stage can be: (a) extremely difficult to perform -mainly due to the system/network complexity, and (b) expensive, both in terms of necessary time and money. In this paper, we present a Java-based software, J-SAEDES, which makes use of discrete-event simulation to: (i) estimate reliability and availability of time-dependent computer systems and networks, (ii) identify those components that play a critical role in the system/network reliability or availability, and (iii) obtain additional information on some system/network performance variables. An application example shows some potential uses of this software.

1 INTRODUCTION

Computer systems and networks usually present hardware and software components that suffer from degradation due to the passage of time and also to intensive use, stress environmental conditions, etc. For that reason, both systems and networks have failures from time to time -i.e.: they occasionally stop working properly (Shooman 2002). A component failure can cause that some of the services that the system/network offers to its users may not be fully available during some time, therefore causing an important economic cost, not only in terms of money but also in terms of personal disturbances and time. When a failure occurs, the affected component may be repaired or replaced by a new one –corrective maintenance-, even when the component failure does not necessarily imply the system/network failure. Additionally, preventive maintenance policies can be employed, i.e.: older components can be replaced by newer ones before their failure occurs (Birolini 2004).

For the context of this paper, we will consider the following definitions: (i) reliability is the probability that a device (system/network or component) will perform its intended function, under operating conditions, for a specified period of time; (ii) availability is the probability that a device, which is operating under some maintenance policy and some concrete conditions, will be operative (performing its intended function) at a certain target-time. Being able to determine system/network reliability and availability (R&A) in the short, medium or long run can be quite useful for system/network managers and industrial engineers in order to ensure data integrity and safety, process or services durability, or even human safety. In all those scenarios, both managers and engineers can benefit from efficient methods and software tools that allow them to obtain information about system/network R&A levels at the design stage.

Most systems and networks can be seen as an aggregation of interconnected components. Therefore, it seems natural to consider the system/network R&A as a function of: (i) the R&A of each individual component, and (ii) the way these components are interconnected (system/network topology). Consequently, predictions at system or network-

level are generally based on the system/network topology and on component-level R&A data (Coit 2000). One of the main targets of the system/network design stage is to provide an assembly of components which, when acting together, will perform satisfactorily for some specified period, either with or without maintenance. When determining system/network reliability, no maintenance policy at component level is considered until a system/network failure is reached and, therefore, only failure-times distributions for each component are needed. Furthermore, when determining system/network availability, maintenance policies at component level -both corrective and preventive- can be considered and, therefore, also repair-times and replacement-times distributions for each component may be needed.

2 USE OF SIMULATION IN R&A ANALYSIS

Real systems and networks use to be highly complex structures characterized by multiple possible states, tens or even hundreds of components, non-series-parallel topologies, non-exponential failure, repair or replacement-time distributions, non-perfect maintenance policies, and existence of dependencies among components. In a context like this, the exclusive use of analytical methods to predict R&A in real systems and networks usually implies the introduction of some unrealistic assumptions that can lead to a reasonable loss of credibility in the model and, consequently, in the final results of the analysis.

The need for implementing simulation models in some reliability scenarios was obvious since the nineties (Marseguerra and Zio 1993), (Goel and Gupta 1997), with some initial experiments in earlier decades (Levin and Kalal 2003). Applications of simulation techniques in the R&A fields allow to model details such as component dependencies, dysfunctional behavior of components, etc. (Dutuit et al 1997), (Labeau and Zio 2002).

Simulation-based techniques have also been proposed to study complex systems and networks availability (Wang and Pham 1997), (Chisman 1998), (Juan and Vila 2002), (Faulin et al 2005). As a matter of fact, during the last years several commercial simulators have been developed to perform R&A analysis over complex systems and networks. Two well-known examples are:

- AvSim+ http://www.isograph-software.com/avsover.htm
- BlockSim http://www.reliasoft.com/BlockSim

Each of these commercial computer programs has it own characteristics, uses its own proprietary simulation algorithms and its own random numbers generator, offering different sets of results and possibilities to the final user.

3 DISCRETE-EVENT SIMULATION WITH JAVA

Java is a general-purpose and object-oriented programming language (Horstmann 2005). Java code is portable –i.e.: it is platform independent, since it runs on the Java Virtual Machine. Furthermore, Java has a rich library of packages which includes different aspects such as graphics, cryptography, networking, database storage, simulation, etc.

Some of the main benefits that Java provides when developing simulation experiments are:

- Being an object-oriented programming language, Java offers us a natural framework for modeling real systems as a collection of objects and events which are relevant to a given task and which are linked in some way.
- Being portable, once a Java simulation program has been implemented and compiled as byte-code, it can run –with few changes, if any- on the most common platforms and operating systems.
- Java allows for multi-threading, that is: multiple execution paths (threads) can be run simultaneously, therefore allowing to model simultaneous activity as it occurs in most real systems.
- Several free or open-source Java packages and frameworks have been already developed to perform discrete-event simulation, among others: DESMO-J (Page and Kreutzer 2005), SSJ (L'Ecuyer 2005), Psim-J (Garrido 2001), and javaSimulation (Helsgaun 2004). These packages contain high-quality Java classes and interfaces which can be easily integrated with the main program to deal with almost any aspect of a simulation experiment, from the random number generation to the management of the simulation clock or the statistical analysis of the results.

4 OUR APPROACH TO THE R&A PROBLEM

Our approach to the problem of determining R&A functions in time-dependent systems and networks at the design stage is centered around the idea of inferring information at system/network level from information regarding: (i) R&A functions at component level, (ii) how components are linked together –i.e. system/network topology, (iii) maintenance policies in use –if any, and (iv) existing dependencies among components –if any (Figure 1). In order to perform this inference process, we use a combination of core statistical results and discrete-event simulation methods to develop an algorithm. This algorithm is eventually implemented as a Java computer program.

In accordance with the previous objective, in the rest of the paper we present and develop the following contents:

- A general framework where both application spectrum and special requirements of our approach are discussed.
- A discrete-event simulation algorithm designed to provide: (a) point and interval estimates for R&A functions at system/network level, (b) identification of "key components" –i.e.: components which are critical for improving overall R&A levels, and (c) information about some system/network performance variables, such as "system time to failure", "system time to repair", "number of system failures per period", "system utilization", etc.
- A Java-based computer program, J-SAEDES, which provides a concrete implementation of the former general simulation algorithm.
- A case study, regarding a wide area computer network, where some potential applications of J-SAEDES are shown.

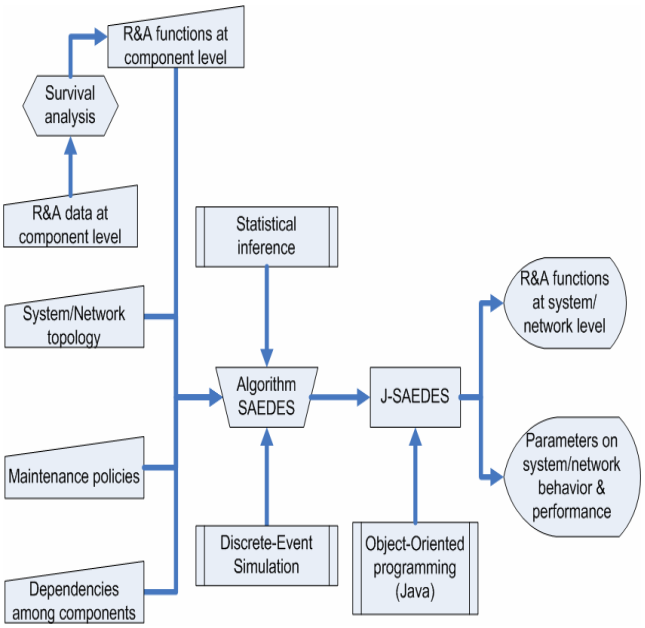

Figure 1: Conceptual approach to the R&A problem

5 FRAMEWORK OF USE

Usually, a lot of data on R&A functions are available at component level. Even if this is not the case, analytical or probabilistic methods -e.g. survival analysis- can be used to obtain these data at component-level (Meeker and Escobar 1998). Therefore, it seems natural to assume the following two principles: (i) for each component, the associated failure-times distribution is known; and (ii) moreover, when maintenance policies are considered at component level, the distributions of repair and replacement times are also known. At this point, it should be noted that repaired or replaced components could not perform "as good as new ones". If that is the case, then our simulation model should also account for these non-perfect maintenance policies. This could be done, for instance, by including a degradation factor for each repaired/replaced component. Arrays can be defined so that new failure-times distributions can be loaded into the simulation model whenever a component is repaired/replaced by a new one not satisfying the "as good as new" property.

In order to be able to infer R&A information at system/network level from data at component level, we also need to know how components are linked together, that is, which logical topology defines the system or network. This is necessary because our simulation algorithm will assume that the system status at any target-time can be derived from the current status of each component at that time, which will be always known during the simulation process. One common way to perform this derivation process is by using minimal paths to describe the system/network topology –a system/network will be operative if, and only if, any of its minimal paths is operative, i.e.: there is at least one path with all its components being operative (Kovalenko et al 1997). Alternatively, minimal cut sets could also be used instead –a system/network will be operative if, and only if, none of its minimal cut sets has failed. At the end, our approach can support any technique that allows determining the system/network status from its components status at any given time, which seems a reasonable requirement. Note that, apart from the previous assumption, there is no other requirement in our model regarding the complexity of the system/network, neither on the number of components nor on its topology. Also, note that no restriction has been imposed neither on the number of different states that a system/network can have nor on the number of different states that any component can reach. In other words, our approach allows to consider multi-state systems/networks composed by multi-state components.

Failure times associated to different components can be statistically correlated. For instance, we can imagine a system/network where several redundant components are all together performing a certain task: failure of one of them will not necessarily cause the system/network failure, but it will probably accelerate other components deterioration process, since these other components will now have to support an extra load. Therefore, a component failure can modify other components failure-times distributions. Repair or replacement-times can also be correlated: each time a component is repaired/replaced, resources that were being employed in its maintenance will get free and, therefore, they could be assigned to maintain other components. Furthermore, correlations can occur among failure and repair or replacement times: if a redundant component is repaired/replaced, it will be able to resume its tasks. This, in turn, will probably imply a reduction in the load supported by other components. Taking care of these subtle dependencies among components seems to be out of the scope of

analytical and static-simulation methods, since these dependencies involve random time-dependent factors. On the contrary, discrete-event simulation allows considering these effects in a natural way. Specifically, in order to account for these dependencies, we propose the use of dependencies arrays. These arrays will contain quantitative information on how each component failure, repair or replacement is changing other components failure, repair or replacement times. One way to do this is by using percentages to indicate deteriorating rates increments or decrements associated to each of the survival components. This way, these effects can be accumulated in the future after new changes in the status of any component –component failure, component repair/replacement, etc. During the simulation process this information can be considered in order to modify, after every event, the tables containing the distributions of failure and repair/replacement-times associated to each component.

6 THE SIMULATION-BASED ALGORITHM

In this section we will provide an overall description of the algorithm proposed as an efficient way to approach the R&A problem in the context of the previously introduced framework. Our algorithm is based on the use of discrete-event simulation to generate multiple instantiations of the system/network life-cycle during some specific period of time (Figure 2).

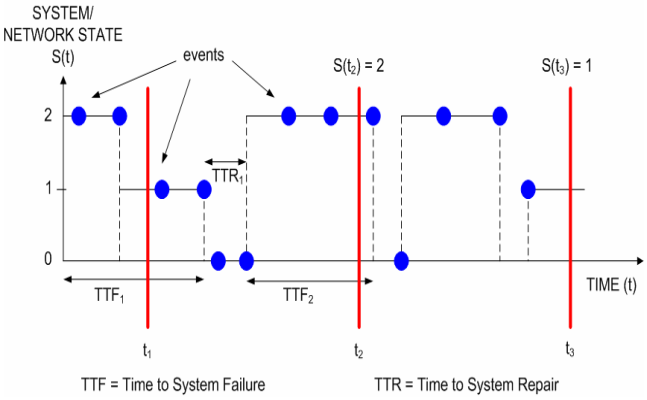

Figure 2: Instantiating a system/network life-cycle

For each instantiation, an event-scheduling list and a simulation clock guide the realization of possible discrete-events that may change the actual state of any component in the system/network (Law 2006). The main events to consider at this level are: (i) a component failure or degradation –if multi-state components are considered, (ii) a component repair, and (iii) a component replacement. Note that some other possible events, such as those regarding variations on available maintenance resources or sudden changes in the working conditions, could also be included in the event-scheduling list as far as there exists enough data to model them by means of statistical distributions. When any of the previous events occurs, the system/network is checked for possible changes in its state –this way, we keep monitoring the state of the system/network at any time and, particularly, at those user-defined target-times where R&A levels need to be calculated. Furthermore, simulation variables and statistics are updated –these statistics can refer to almost any information regarding the system internal behavior and performance, including: counts of failures per component, indices to measure system/network sensitivity to component failures, distribution of system/network failure or repair-times, total time that the system/network is available or being repaired, mean time between system/network failures or repairs, etc. Finally, some additional actions will occur to deal with the event that has taking place. For instance, if the event is a component failure, then a change in the component state is registered, a repair-time for that component is scheduled –according with available maintenance resources, and dependencies matrices are checked out to account for possible secondary effects of this failure over other components' failure or repair-times. On the contrary, if the event is a component repair or replacement then a change in the component state is registered, employed maintenance resources are liberated, a new failure-time for the component is scheduled –considering non-perfect maintenance policies if necessary, and dependencies matrices are checked out again to account for possible secondary effects of this event over other components' failure or repair-times. This process will continue until the simulation clock reaches the end time of the period of interest.

Using simulation variables and statistics, several parameters regarding the system/network behavior and performance can be defined. At the end of each instantiation process, a random observation for each of these parameters is obtained. This instantiation process can be replicated a certain number of times, which will provide us with a random sample of observations for each parameter of interest. These observations, in turn, can be statistically analyzed in order to obtain valuable information about the system/network behavior and performance. In particular, since we have obtained a random sample of observations on the system/network state at each target-time, we can use statistical inference techniques to generate point and interval estimates for system/network R&A levels at each target-time. The key idea to do this is explained below: for the multi-state system/network under study, consider the set of probabilities associated to each possible state. Note that each of these probabilities can be seen as a success probability in a multinomial distribution. Therefore, an unbiased and consistent point estimate for these probabilities can be easily obtained as the quotient between the number of observations for the corresponding state and the total number of instantiations. Furthermore, using the Central Limit Theorem (Ross 2001), confidence intervals for these probabilities can also be calculated. This way, both point and inter-

val estimates can be obtained for R&A levels at each target time, since R&A are related to the probability of the system being in an operative status at a particular time –the precise definition of "being operative" could vary depending on the specific characteristics and functionality of the system/network under study and also on managers criteria. Note that it may be necessary to compute several hundreds of thousands or even millions of instantiations to obtain precise interval-based estimations. In some extreme cases, variance reduction techniques or parallel simulation techniques should be considered as a way to accelerate the computational process. Finally, regarding the simulation inputs, note that we could provide the desired maximum error of estimate as an alternative to provide the number of instantiations to generate. This way, instantiations will still being generated until the desired precision level is eventually reached.

7 SOFTWARE IMPLEMENTATION

Developing a Java-based implementation of the former algorithm is not a trivial task, since this can be done using different approaches regarding: (i) input/output options, (ii) use of auxiliary libraries -specially those associated to random number generation and to statistical analysis, (iii) programming approaches and classes design, (iv) use of random variance techniques or parallel processing and, which maybe be even more important, (v) levels of accuracy and effectiveness –as an example of this, to perform mathematical operations the use of the `java.lang.StrictMath` package is usually preferred over the use of the more classical `java.lang.Math` package. It is in this context where the role of open-source programming communities can be essential. Working collaboratively online, these communities can develop low-cost, efficient, usable and flexible reliability/availability software based on the ideas presented in this paper.

The current version of J-SAEDES follows a modular structure (Figure 3) with the following main classes:

- Class `saedesIn`: This class is responsible for providing the inputs for the simulation experiment, which are user-defined. The inputs can be entered using different file formats such as txt, xls or xml.
- Class `saedesOut`: This class contains direct simulation results as well as post-analysis results. The resulting file can have different formats, which facilitates exportation to external data-analysis programs.
- Class `saedesRndVar`: This class manages the generation of pseudo-random numbers and random variates. The election of high-quality generation algorithms is vital for adding credibility to the simulation results, specially in long simulations (L'Ecuyer 2001)
- Class `saedesPerSim`: This class contains the kernel of the J-SAEDES application. It encapsulates the algorithm described in the previous section.
- Class `saedesStats`: This class contains methods that perform all the necessary statistical calculations –descriptive statistics, confidence intervals, etc.
- Class `saedesReport`: This class allows to generate formatted reports from the simulation results.

Several tests have been carried out using the former implementation. Whenever possible, outputs obtained with J-SAEDES have been checked against those provided by existing commercial programs. In every case, results have shown to be convergent, which contributes to verify our program and to validate the proposed algorithm. The following section includes an case study that show some potential applications of our approach.

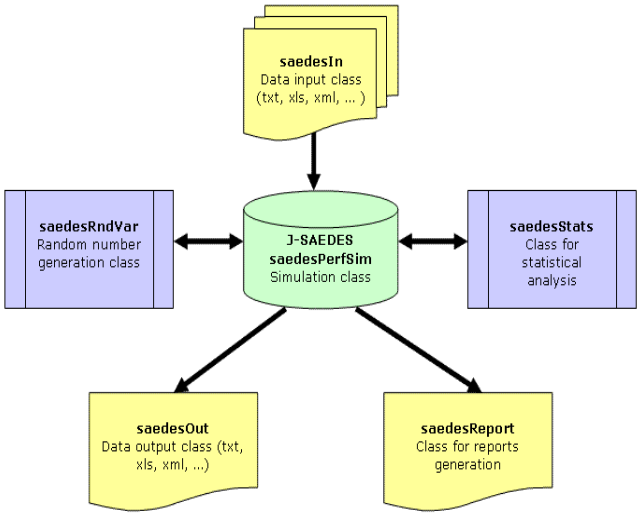

Figure 3: J-SAEDES architecture

8 CASE STUDY: A WIDE AREA NETWORK

Consider the Wide Area Network (WAN) whose physical topology is represented in (Figure 4), where:

- Each cylindrical symbol (numbers 1 to 20) represents an ITN (Information Transport Node) consisting in a set of routers and/or high-level switches
- Each city symbol (labeled "NW EUROPE", "NE EUROPE", "SW EUROPE" and "SE EUROPE") represents an enterprise Local Area Network (LAN) located at four different cities. LANs users make use of the WAN to share information among them

- Each edge represents an optic-fibre cable (or similar transmission medium) with enough bandwidth to support an intensive use of the WAN

For our illustration purposes, we will consider that an ITN has failed whenever it cannot be able to forward the information received. The failure can be caused by multiple factors (both internal and external): ageing of the devices that composes the ITN, sudden failure of some electronic component, external hacker attacks, etc.

Moreover, the WAN is considered to be working correctly whenever all the LANs (end users) can communicate among them, i.e., whenever there exists some paths that connect all the LANs (observe that this can happen even when some ITNs have failed). Therefore, a failure of the system will take place whenever any of the four LANs gets isolated from the rest.

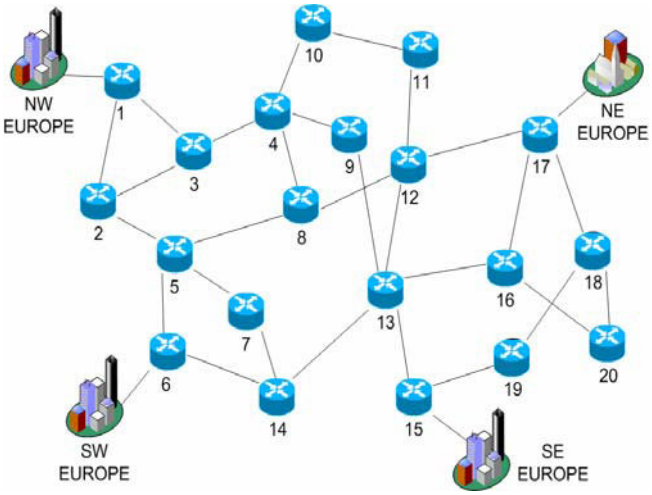

Figure 4: An enterprise Wide Area Network

Table 1 shows the statistical distributions of the failure and repair times (in years) associated to each ITN component. As stated before in this paper, this information is assumed to be known beforehand –note that it might be necessary to use survival analysis techniques in order to obtain it.

Obtaining the minimal path decomposition for the network of this case study takes some analysis (Faulin et al 2007). Finally, 23 paths that were identified as the only minimal paths associated to this network.

Under these conditions, which will be the availability of this network at the following ten target times (in years): $t_0 = 0.1, 0.2, ..., 1.0$? To answer this question, we have employed the J-SAEDES program.

After employing less than a minute to perform the specified one-million iterations (using a standard notebook with a Pentium IV processor at 2.8GHz and 512MB of RAM), has provided the point and interval estimates -at a 99% confidence level- for the network availability function at each of the specified target-times (Table 2)

TABLE 1: FAILURE AND REPAIR TIMES FOR EACH COMPONENT

	Failure Times			**Repair Times**		
ITN	***Distribution***	***P1***	***P2***	***Distribution***	***P1***	***P2***
1	Weibull	1.32	2.13	Weibull	1.32	2.13
2	Weibull	1.34	2.12	Weibull	1.34	2.12
3	Weibull	1.33	2.11	Weibull	1.33	2.11
4	Weibull	1.00	1.20	Weibull	1.00	1.20
5	Weibull	1.22	2.01	Weibull	1.22	2.01
6	Weibull	1.20	1.22	Weibull	1.20	1.22
7	Weibull	1.20	2.01	Weibull	1.20	2.01
8	Weibull	1.21	2.09	Weibull	1.21	2.09
9	Weibull	1.31	2.03	Weibull	1.31	2.03
10	Weibull	1.24	2.15	Weibull	1.24	2.15
11	Weibull	1.27	2.03	Weibull	1.27	2.03
12	Weibull	1.02	1.82	Weibull	1.02	1.82
13	Weibull	1.21	2.25	Weibull	1.21	2.25
14	Weibull	1.20	1.72	Weibull	1.20	1.72
15	Weibull	1.12	2.33	Weibull	1.12	2.33
16	Weibull	1.14	2.22	Weibull	1.14	2.22
17	Weibull	1.34	2.21	Weibull	1.34	2.21
18	Weibull	1.10	1.64	Weibull	1.10	1.64
19	Weibull	1.32	2.12	Weibull	1.32	2.12
20	Weibull	1.23	1.29	Weibull	1.23	1.29

TABLE 2: AVAILABILITY RESULTS FROM J-SAEDES

Target-time	**Network Availability Estimates**		
	Point	***99% Confidence Interval***	
0.1	0.888278	0.887467	0.889089
0.2	0.749102	0.747985	0.750219
0.3	0.611353	0.610097	0.612609
0.4	0.486972	0.485685	0.488259
0.5	0.381589	0.380338	0.382840
0.6	0.295792	0.294616	0.296968
0.7	0.227959	0.226878	0.229040
0.8	0.175477	0.174497	0.176457
0.9	0.135686	0.134804	0.136568
1.0	0.105482	0.104691	0.106273

Additional results about the network behavior have been also provided by J-SAEDES, as described in the algorithm section.

9 CONCLUSIONS

A Java-based computer program, J-SAEDES, has been presented in this paper. This program implements a general methodology, based on core statistical results, and a dis-

crete-event simulation algorithm to provide point and interval estimates for the reliability and availability values of time-dependent systems and networks. The program can consider details such as multi-state systems/networks, dependencies among failure and repair-times, or non-perfect maintenance policies.

J-SAEDES can be very helpful for system managers and engineers in order to improve system/networks reliability or availability levels, since it is able to provide useful information about system/network reliability/availability and can be applied in most situations where analytical methods are not well suited. In fact, a simulation program like this, should be used even when analytical methods are applicable, since simulation provides additional information about critical components -from an availability point of view- and about the system internal behavior.

ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Education under grant TSI2005-08225-C07-05. We would also like to gratefully acknowledge the support received from the DPCS research group of the IN3-UOC.

REFERENCES

Birolini, A. 2004: *Reliability Engineering*. Springer-Verlag.

Chisman, J. 1998. Using Discrete Simulation Modeling to study Large-Scale System Reliability/Availability. *Computers and Operations Research*, vol. 25, pp. 169-174.

Coit, D. 2000: System Reliability Prediction Priorization Strategy. In *2000 Proceedings Annual Reliability and Maintainability Symposium*, Jan 24-27, Los Angeles, CA, 2000, pp. 175 – 180.

Dutuit, Y.; Châtelet, E.; Signoret, J. and P. Thomas. 1997. Dependability Modelling and Evaluation by Using Stochastic Petri Nets: Application to Two Test Cases. *Reliability Engineering and System Safety*, vol. 55, pp. 117-124.

Faulin, J.; Juan, A.; Bargueño, V. and C. Serrat. 2005: SAEDES++: Determining Complex System Availability via Simulation. In *Proceedings of the 2005 Winter Simulation Conference*, pp. 1463-1472.

Faulin, J.; Juan, A.; Serrat, C. and V. Bargueño. 2007: Using simulation to determine reliability and availability of telecommunication networks. *European Journal of Industrial Engineering*, vol. 1, issue 2, pp. 131-151.

Garrido, J. 2000. *Object-Oriented Discrete-Event Simulation with Java*. Kluwer Academic.

Goel, L. and R. Gupta. 1997. A Windows-based Simulation Tool for Reliability Evaluation of Electricity Generating Capacity. *International Journal of Engineering Education*, vol. 13, pp. 347-357.

Helsgaun, K. 2004. *Discrete Event Simulation in Java*. Department of Computer Science, Roskilde University, Denmark.

Horstmann, C. 2005. *Big Java*. Wiley.

Juan, A. and A. Vila. 2002: SREMS: System Reliability using Monte Carlo Simulation with VBA and Excel. *Quality Engineering*, 2002; vol. 15, pp. 333-340.

Kovalenko, I.; Kuznetsov, N. and P. Pegg. 1997. *Mathematical Theory of Reliability of Time Dependent Systems with Practical Applications*. John Wiley & Sons.

L'Ecuyer, P. 2001: Software for Uniform Random Number Generation: Distinguishing the Good and the Bad. In *Proceedings of the 2001 Winter Simulation Conference*.

L'Ecuyer, P. 2005. Simulation in java with SSJ. In *Proceedings of the 2005 Winter Simulation Conference*, pp. 611-620.

Labeau, P. and E. Zio. 2002. Procedures of Monte-Carlo Transport Simulation for Applications in System Engineering. *Reliability Engineering and System Safety*, vol. 77, pp. 217-228.

Law, A. 2006. *Simulation Modeling & Analysis*. New York: McGraw-Hill.

Levin, M. and T. Kalal. 2003. *Improving Product Reliability: Strategies and Implementation*. New York: John Wiley & Sons.

Marseguerra, M. and E. Zio. 1993: Nonlinear Monte Carlo Reliability Analysis with Biasing towards Top Event. *Reliability Engineering and Systems Safety*, vol. 40, pp. 31-42.

Meeker, W. and L. Escobar. 1998. *Statistical Methods for Reliability Data*. John Wiley & Sons.

Page, B. and W. Kreutzer. 2005. *The Java Simulation Handbook*. Shaker Verlag.

Ross, S. 2001. *Simulation*. Academic Press.

Shooman, M. 2002: *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis, and Design*. John Wiley & Sons.

Wang, W. and H. Pham. 1997. Survey of Reliability and Availability Evaluation of Complex Networks using Monte Carlo Techniques. *Microelectronics Reliability*, vol. 37, pp. 187-207.

AUTHOR BIOGRAPHIES

ANGEL A. JUAN is an associate professor of Applied Statistics and Simulation at the Open University of Catalonia (Barcelona, Spain). He is also an assistant professor of Applied Statistics at the Technical University of Catalonia (Spain). He holds a PhD in Industrial Engineering and a MS in Applied Mathematics. His research interests include discrete-event simulation and mathematical e-learning. He

is also a member of INFORMS. He can be contacted at <angel.alejandro.juan@upc.edu>.

JAVIER FAULIN is an associate professor of Statistics and Operations Research at the Public University of Navarre (Pamplona, Spain). He holds a PhD in Economics and a MS in Applied Mathematics His research interests include logistics, vehicle routing problems and simulation modeling and analysis, especially techniques to improve simulation analysis in practical applications. He is also a member of INFORMS. His e-mail address is <javier.faulin@unavarra.es>.

JOAN M. MARQUÈS is an associate professor at the Open University of Catalonia (Barcelona, Spain). His research interests include design of scalable and cooperative Internet services and applications, peer-to-peer systems, and distributed computing. Marquès has a PhD in Computer Science from the Technical University of Catalonia (Spain). He is a member of both the ACM and the Kaleidoscope Network of Excellence. He can be contacted at <jmarquesp@uoc.edu>.

MATEO SORROCHE is an assistant professor of Electronic Technology at the Technical University of Catalonia (Manresa, Spain). He holds a BSc in Computer Science and a MS in Software Engineering. His research interests includes graphical systems for simulation and control. His e-mail address is <mateo.sorroche@upc.edu>.