Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B Armstrong, and J. A. Joines, eds

SAEDES++: DETERMINING COMPLEX SYSTEM AVAILABILITY VIA SIMULATION

Javier Faulin

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

Angel A. Juan
Carles Serrat

Department of Applied Mathematics I
Av. Doctor Marañón, 44-50
Technical University of Catalonia
Barcelona, 08028, SPAIN

Vicente Bargueño

Department of Applied Mathematics I
ETS Ingenieros Industriales
Universidad Nacional de
Educación a Distancia
Madrid, 28080, SPAIN

ABSTRACT

Complex systems are everywhere among us: telecommunication networks, computers, transporting vehicles, and electrical appliances are well known examples. Designing reliable systems and determining their availability are both very important tasks for managers and engineers, since reliability and availability have a strong relationship to other concepts such as quality and safety. Furthermore, these tasks are extremely difficult, due to the fact that analytical methods can become too complicated, inefficient or even inappropriate when dealing with sophisticated systems. In this paper we present the basic ideas behind a simulation-based method, called SAEDES, which can be very useful when determining availability for a wide range of complex systems. The method is implemented in C/C++ using two different algorithms, SAEDES_A1 (component-oriented) and SAEDES_A2 (system-oriented). Two case-studies are introduced and analyzed using both algorithms, which allows us to compare the associated results.

1 INTRODUCTION

Physical devices suffer from degradation, not only due to the passage of time, but also due to an intensive use. This process can also be observed in intangible products such as computer software. For instance, network operating systems tend to stop working properly from time to time and, when that happens, they need to be reinstalled or, at least, restarted, which means that the host server will stop being available during some time (Birolini 2004, Shooman 2002) At the end, if no effective maintenance policies are taken, any product (component or system, hardware or software) will fail, meaning that it will stop being operative, at least as intended.

Reliability is often defined as the probability that a system or device will perform its intended function, under operating conditions, for a specified period of time (Pham 2003). On the other hand, availability can be defined as the probability that a system or device, according to some maintenance policy and some operating conditions, performs its intended function at a certain time.

Maintenance policies are applied to a lot of systems, in a way that when one component goes down, it is repaired or a new one is substituted –even when the component failure does not mean the global system failure. In such situations, it is obvious that knowing the system availability in the short, medium or long run will be useful information for engineers and managers in order to ensure data integrity and safety, process or services durability, or even human safety.

In all those scenarios, managers and engineers can benefit from efficient methods and software tools that help them to get information about system availability and help them to increase it.

2 SIMULATION IN AVAILABILITY STUDIES

Reliability and availability of time-dependent complex systems is a research area with applications not only to engineering but also to experimental and social sciences (Collet 2003, Levin and Kalal 2003, Traub 1994).

Different analytical approaches can be used in order to calculate the exact reliability of a time-dependent complex system (Kovalenko et al. 1997). Unfortunately, when the system is highly complex (that is, having many components with a non-series-parallel logical structure or a non-exponential failure-time distribution), it can become extremely difficult or even impossible to obtain its exact reliability at a given target time. Similar problems arose when trying to determine the exact availability at a given target time for systems subject to maintenance policies. As some authors point out (Bajenescu 1998, Goel and Gupta 1997, Billinton and Wang 1999, Chisman 1998), in those situations only simulation techniques, such as Monte Carlo Simulation (MS) and Discrete Event Simulation (DES), can be useful to obtain estimates for reliability and availability parameters.

One of the first good ideas and algorithms on the use of MS techniques to estimate system reliability can be found in Kamat and Riley (1975). MS techniques have also been proposed to study complex systems availability (Wang and Pham 1997). In fact, during the last years, several commercial simulators have been developed to study the reliability and availability of complex systems (Willis 2000). These simulators include: AvSim+, BlockSim, MEADEP, RAM Commander, RAPTOR, Relex RBD, SPAR and TIGER.

In this paper, a new method, called SAEDES, is presented. The ultimate objective of this method is to determine (estimate) a complex system availability using the following information (which is assumed to be known): (a) system logical structure, and (b) failure-times and repair-times distributions for each component. The method is implemented using two different algorithms, SAEDES_A1 (which uses MS and can be considered as component-oriented in the sense that it is based on the generation of each component history) and SAEDES_A2 (which uses DES and can be considered as system-oriented in the sense that it is based on the generation of system history). Both algorithms are inspired on ideas introduced by Singh and Mitra (1995) and Juan and Vila (2002).

3 WHICH SYSTEMS ARE WE CONSIDERING?

The method presented in this paper, SAEDES, has been designed to deal with any kind of logical or physical system that meets some general criteria. The hypotheses made by SAEDES are common assumptions in availability studies and they are less restrictive than the ones used when applying analytical methods.

Specifically, when studying the availability of systems, which are subject to maintenance policies (i.e. it will be considered components repair or substitution while the system still is working properly), SAEDES will make the following main assumptions:

1. **Two-state systems:** at any given time, the system will be either operational (working properly) or not. Observe that the exact definition of "being operational" is up to the system managers, since it will vary depending of the system and its environmental circumstances
2. **Coherent systems:** the analyzed system is assumed to be coherent, in other words: if every component is operative the system will be operative, if no component is operative the system will not be operative, and a positive status change in a component (that is, from inoperative to operative) cannot cause a negative status change in the system (that is, the system will not change its status from operative to no operative)
3. **Minimal paths decomposition:** the system logical structure is known and it can be expressed in the form of minimal paths (Kovalenko et al. 1997)
4. **Component failure-times and repair-times distributions:** for each component, its associated failure-times and repair-times distributions are perfectly known (i.e. both the statistical distribution family and exact parameters are known)
5. **Maintainability policy:** system is under a continuous inspection policy, that is, any failure will be detected as soon as it will appear
6. **Perfect reparations or substitutions:** when a component fails, it is repaired or substituted by a new one; in any case, the result is as if a new component has been placed
7. **Failure-times and repair-times independence:** the failure-times associated to one specific component are independent from the failure-times associated to any other component; the same holds true for repair times.

Assumptions (1) to (4) guarantee that there is enough information to study the system reliability. Assumption (3) often requires a detailed analysis of logical relationships among components. In this sense, simulation algorithms have been proposed to find out the minimal path decomposition of a complex system (Lin and Donaghey 1993). In the assumption (4) context, statistical methods such as accelerated live tests (Meeker and Escobar 1998) and data fitting techniques (Leemis 2003) are usually required. Assumptions (5) and (6) are not restrictive in the sense that they could be relaxed, if necessary, by adapting the algorithms of the method.

Finally, assumption (7) is the most restrictive one and it may require considering some abstraction levels in the system decomposition. For example, if the system was a PC it could be necessary to join several pieces, such as the microprocessor and its associated fan, into just one component. Otherwise, it could not be possible to assume independence among components failure-times or among components repair-times.

4 MATHEMATICAL FUNDAMENTALS

SAEDES method and algorithms make use of several mathematical concepts and techniques. Specifically, the method is based on:

- **System availability theory:** system reliability and availability concepts, including minimal paths theory (Barlow and Proschan 1996, Hoyland and Rausand 1994, Kovalenko et al. 1997, Pham 2003)
- **Simulation techniques:** data fitting, pseudo-random number generation, event treatment, and

variance reduction methods (Banks 1998, Chung 2004, Law and Kelton 2000, L'Ecuyer 2002, Wang and Pham 1997)

- **Probability and statistical concepts:** probability theory, descriptive statistics and inference techniques (Ross 1996).

Let us explain the SAEDES driving data. Given a fixed instant $t_0 \geq 0$, the main target of the simulator is to estimate the system reliability at that time, i.e.:

$$p(t_0) = P\left(\phi\left(\mathbf{X}(t_0)\right) = 1\right)$$

where $\mathbf{X}(t_0)$ is the status vector of the system at t_0 (it describes the actual status of each component at t_0), and $\phi\left(\mathbf{X}(t_0)\right)$ is the binary status function (value will be 1 if the system is operational at t_0, and 0 otherwise).

Considering the system has two possible states at any given time (operational and non-operational), the system status at t_0 can be interpreted as a binary variable following a Bernoulli distribution, being $p(t_0)$ the probability of "success" (understanding success as the fact that the system is operational at t_0), i.e.:

$$Y = Y(t_0) = \phi\left(\mathbf{X}(t_0)\right) \sim Be\left(p(t_0)\right)$$

In that situation, if we were able to obtain, using some simulation algorithms, m random and independent observations, $Y_1, Y_2, ..., Y_m$, from the binary variable Y, which represents the system status at t_0, we know that the sum of those variables will be a new one following a binomial distribution of parameters m (number of proofs) and $p(t_0)$ (probability of "success" in each proof), i.e.:

$$\sum_{i=1}^{m} Y_i \sim Bi\left(m, p(t_0)\right)$$

At this point, it is known that the maximum likelihood estimator for $p(t_0) = \phi\left(\mathbf{X}(t_0)\right)$ is given by the sample mean:

$$p' = \overline{y}_i = \frac{\sum_{i=1}^{m} y_i}{m}$$

This is an unbiased estimator, since:

$$E[p'] = \frac{1}{m} E\left[\sum_{i=1}^{m} y_i\right] = \frac{1}{m} \sum_{i=1}^{m} E[y_i] = p(t_0)$$

Furthermore, the law of large numbers says that the former estimator will tend to improve as the number of observations gets larger, more concisely:

$$P\left(\lim_{m \to \infty} p' = p(t_0)\right) = 1$$

Observe that, apart from obtaining a point estimate, it is also possible to obtain confidence intervals for $p(t_0)$. In effect, the central limit theorem asserts for large values of m that:

$$\sum_{i=1}^{m} Y_i \sim N\left(m \cdot p(t_0), \sqrt{m \cdot p(t_0) \cdot \left(1 - p(t_0)\right)}\right)$$

i.e.:

$$p' \sim N\left(p(t_0), \sqrt{\frac{p(t_0) \cdot \left(1 - p(t_0)\right)}{m}}\right)$$

Therefore, a confidence interval for $p(t_0)$ with a $1-\alpha$ confidence level will be:

$$p' \pm z\left(\frac{\alpha}{2}\right) \cdot \sqrt{\frac{p(t_0) \cdot \left(1 - p(t_0)\right)}{m}}$$

where $z\left(\frac{\alpha}{2}\right)$ is the $1-\frac{\alpha}{2}$ percentile in a standard normal distribution (in practice, since $p(t_0)$ is not known, its value will be substituted by the point estimate p').

The method just described here is based on core statistical concepts. Therefore, it is obvious that the key point of SAEDES consists in developing of a simulation-based algorithm, which provides us with the m random and independent observations from the variable Y. In the next sections, two such algorithms are explained, SAEDES_A1 and SAEDES_A2. In a sense, they are alternative algorithms that can be used to obtain the desired observations, both having special properties that will be discussed in the case-study sections. It is important to observe, though, that SAEDES_A2 not only provides information on the system availability at a specified target time, but it will also offer other system parameters estimations: percentage of time that the system has been available, number of system failures during the experimentation period, mean time to system failure (MTTF) and mean time to system repair (MTTR).

The implementation of both SAEDES_A1 and SAEDES_A2 algorithms as computer programs has a lot of technical details, both of mathematical and programming nature. For space reasons, only the general ideas behind these algorithms will be presented in this paper.

5 ALGORITHM SAEDES_A1

As stated before, a key part of the SAEDES method is to obtain m random and independent observations from the binary variable Y. To do that task, we have developed the algorithm SAEDES_A1 (Figure 1), which can be summarized in four steps as follows:

1. **Step 1:** For each of the n components, use simulation to generate its history until the target-time t_0 (in this sense, it can be said that this algorithm is component-oriented). This can be achieved generating random failure-times, T_i, and random repair times, D_i, for the i*th* component (see Figure 2). Then, determine each component availability at t_0, $X_i(t_0)$
2. **Step 2:** Using (a) observations $X_i(t_0)$, $i=1,2,...,n$, and (b) system structure (minimal paths decomposition), determine system status at t_0, $\phi(\mathbf{X}(t_0))$. In order to do that, it will be necessary to make use of system minimal paths: system will be operative at t_0 if, and only if, any of its minimal paths is operative at t_0 (i.e. each and every component on that path is operative at t_0)
3. **Step 3:** If $\phi(\mathbf{X}(t_0))=1$ then assign $Y=1$ (a "success" has taken place). On the other hand, if $\phi(\mathbf{X}(t_0))=0$, assign $Y=0$
4. **Step 4:** Repeat m times steps 1 to 3 (that is, repeat those steps as many times as the number of desired observations).

6 ALGORITHM SAEDES_A2

Algorithm SAEDES_A2 represents an alternative way to obtain the m random and independent observations from the binary variable $Y=\phi(\mathbf{X}(t_0))$ stated in the SAEDES method. This new algorithm is absolutely different to SAEDES_A1, which was component-oriented in the sense that it was based on the individual generation of each component history (this orientation makes that algorithm specially suitable for parallel simulation, that is, multiple computers can be used simultaneously to perform the simulation study). On the other hand, algorithm SAEDES_A2 is system-oriented since it is based on the generation of the system history (Figure 3).

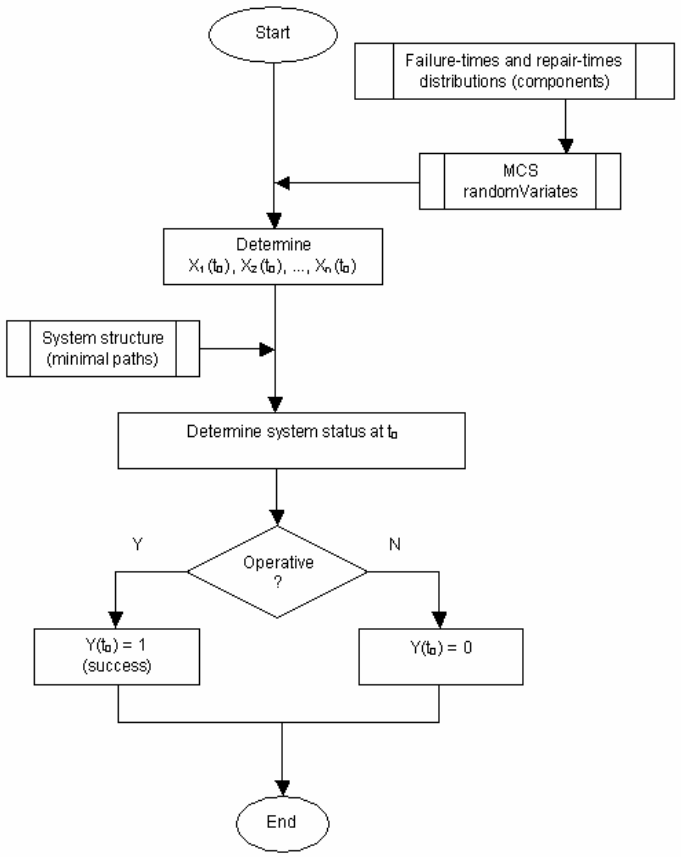

Figure 1: Flow Chart for SAEDES_A1

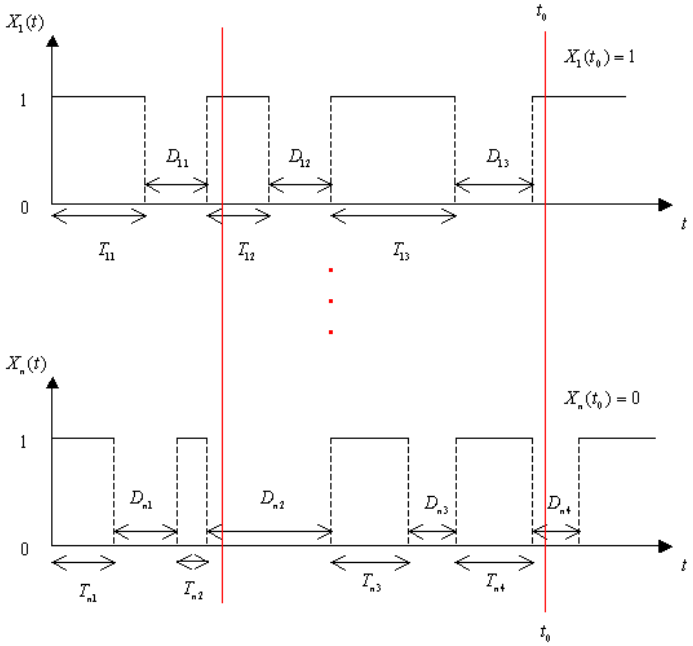

Figure 2: SAEDES_A1 is Component-Oriented

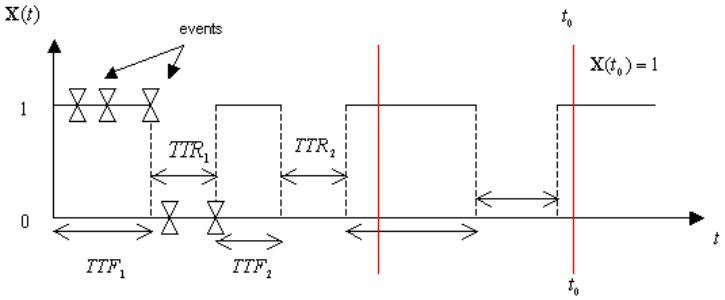

Figure 3: SAEDES_A2 is System-Oriented

This new orientation complicates the simulation model somewhat, and makes necessary the use of discrete-event simulation techniques, but it allows us to obtain much more information about the system (in fact, the model details could be increased such in a way that we could obtain detailed information about the system behavior).

Algorithm SAEDES_A2 uses discrete-event simulation techniques with two possible events, a component failure and a component repair. This algorithm can be summarized in the following steps (Figure 4):

1. **Step 1:** Start simulation variables (that is to say: simulation clock, each component status, system status and statistics)
2. **Step 2:** Assign a random failure time to each component and deactivate each component repair-time
3. **Step 3:** Determine next event to happen and get the following information about it: type (component failure or component repair), time when it will take place (next-event time), and component affected by it (next-event component)
4. **Step 4:** Using information from previous step, update the variable that represents the total time that the system has been available
5. **Step 5:** If there are any target-times between clock-time and next-event time, assign a value of 1 to those target times in which the system was available, and a value of 0 to those target times in which it was not
6. **Step 6:** Update next-event component status. If the event is a component failure, generate a random repair time for that component and deactivate its associated failure-time. Conversely, if the event is a component repair, generate a random failure-time for that component and deactivate its associated repair-time
7. **Step 7:** Verify if the next-event will cause a system status change (that is, if the system will stop being available or vice versa). If that is the case, update system status and statistics variables
8. **Step 8:** Update simulation clock
9. **Step 9:** Repeat steps 3 to 8 until terminating condition is met (iteration will finish once simulation clock passes all target times)
10. **Step 10:** Register current iteration values
11. **Step 11:** Repeat *m* times steps 1 to 10 (that is, repeat those steeps as many times as the number of desired observations).

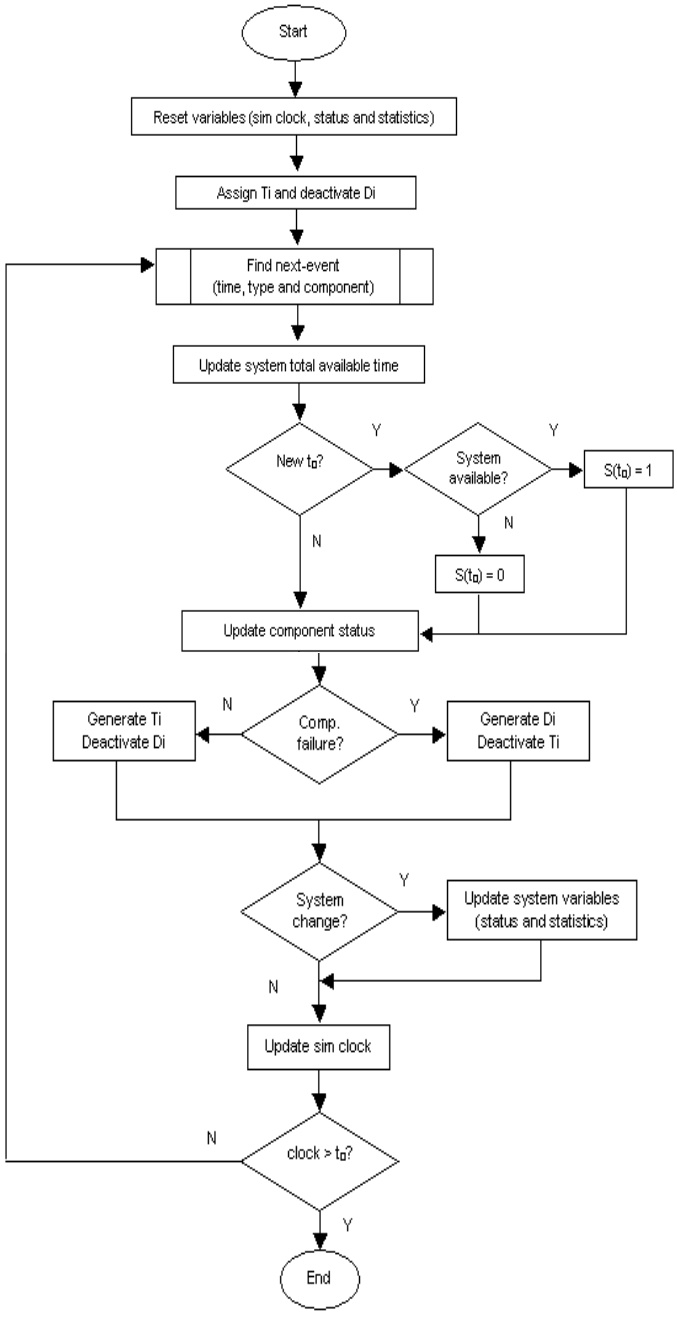

Figure 4: Flow Chart for SAEDES_A2

7 SOFTWARE IMPLEMENTATION

As in any other simulation experiment, two main objectives are the next ones: (a) to validate the accuracy and effectiveness of the proposed method, SAEDES, and (b) to provide a computer version of the mathematical model that could be efficiently used by system managers and engineers. In order to attain those targets, we have developed two C/C++ programs, SAEDES+ -which makes use of algorithm SAEDES_A1, and SAEDES++ -which makes use of algorithm SAEDES_A2.

As Figure 5 shows, the programs SAEDES+ and SAEDES++ have a modular structure including: a kernel, which takes care of the simulation model, the `randomVariates` library -which is a random variates generator that includes a well-tested and long-period pseudo-random numbers generator (L'Ecuyer 2002), and the `stats` library -which performs all the required statistical operations (descriptive, confidence intervals, etc.) so that the programs do not need to make use of any other additional software.

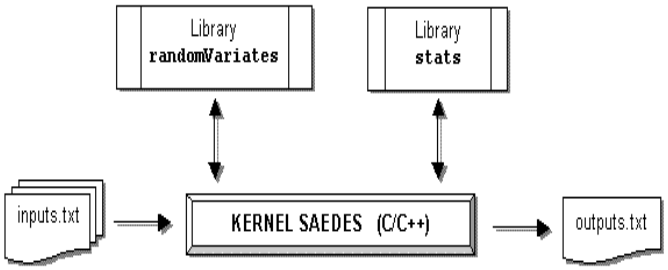

Figure 5: Modular Structure of SAEDES++

The program inputs are entered by the user using three simple `txt` files, and after running the simulation, the program provides a single `txt` file with the results. The three inputs files must contain the following information (i.e. these are the model inputs):

1. `saedes_inputs_first.txt`

   - **Number of iterations** to run (more iterations implies not only more accurate estimates, but also an increase in computational costs such as simulation length and memory resources)
   - **Number of components** in the system
   - **Number of target times** (times at which system reliability has to be determined)
   - **Time interval** (step) between consecutive target times
   - **Number of paths** in the logical structure
   - **Seed** of the simulation

2. `saedes_inputs_second.txt`

   - **Failure-times distribution** associated with each component
   - **Repair-times distribution** associated with each component
   - **Length** of each path

3. `saedes_inputs_third.txt`

   - **System path composition** (array of components that make up each path).

On the other hand, once the simulation has been run, the outputs file, `saedes_outputs.txt`, provides the following information about the system being modeled (i.e.: these are the model outputs):

a. **Point and interval estimates** for the system availability at different target times.

b. **Expected values** for: total time that the system is available, mean time to failure, mean time to repair, and number of failures during the simulation experiment.

Several tests have been carried out using both programs, and systems with different levels of complexity have been studied within those tests. Results obtained with one algorithm have been compared with the ones obtained using the other. All tests have given satisfactory results in the sense that both algorithms provide convergent results – which, in turn, let us significantly increase the level of confidence in the SAEDES method, since the two algorithms are based on two completely different approaches.

The following section includes two of the experiments that have been performed to validate the method and verify the programs.

8 STUDY CASE 1

A simple system is shown in Figure 6. The system has seven components and three minimal paths. Table 1 contains the failure-time distribution and repair-time distribution for each of the components (for simplicity, it is assumed that failure times and repair times follow exactly the same distribution).

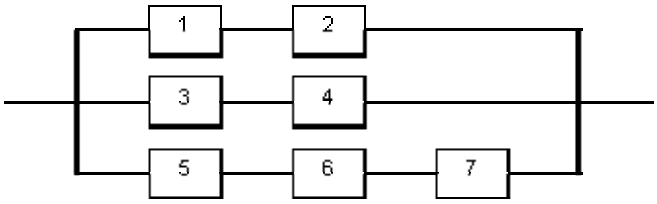

Figure 6: System for Case 1

Table 1: Failure and Repair Times Distributions for Case 1

Component	Distribution	Parameter 1	Parameter 2
1	Weibull	1.32	2.13
2	Weibull	1.34	2.12
3	Weibull	1.33	2.11
4	Exponential	1.2	--
5	Weibull	1.22	2.01
6	Exponential	1.22	--
7	Weibull	1.20	2.01

We have used both SAEDES+ and SAEDES++ to obtain estimates for the system availability at ten different target times, $t_0 = 0.5,\ 1.0,\ 1.5,\ 2.0,\ ...,\ 5.0$.

Figure 7 shows results obtained with SAEDES+ (that is, using algorithm SAEDES_A1) after running 500,000 iterations:

```
saedes_outputs.txt - Bloc de notas
Archivo  Edición  Buscar  Ayuda
SIMULATION OUTPUTS:

algorithm: SAEDES_A1
sim lenght:  58 seconds

target  availab.          99% CI
1:      0.955832    (0.955084 , 0.956580)
2:      0.813418    (0.811999 , 0.814837)
3:      0.673722    (0.672014 , 0.675430)
4:      0.579418    (0.577620 , 0.581216)
5:      0.527408    (0.525589 , 0.529227)
6:      0.503436    (0.501615 , 0.505257)
7:      0.495714    (0.493893 , 0.497535)
8:      0.495990    (0.494169 , 0.497811)
9:      0.497032    (0.495211 , 0.498853)
10:     0.501598    (0.499777 , 0.503419)
```

Figure 7: SAEDES+ Outputs for System 1

It follows from the outputs file (Figure 7) that the program SAEDES+ has been able to provide us with point and interval estimates for the system availability at the ten different target times. So, for instance, at $t_0 = 0.5$ the estimated system availability is 0.9558. It can also be observed that the system availability tends to reach a stable value of 0.5 This should not be surprising if we consider the symmetry introduced in the system by considering that both repair times and failure times were following the same distributions and, also, by the fact that no component is repeated in any two different paths.

The same system has been simulated with SAEDES++ (that is, using algorithm SAEDES_A2). Again, 500,000 iterations have been run. Figure 8 shows the basic outputs file, and Table 2 compares results from both programs.

From Table 2, it seems clear that both algorithms are providing equivalent estimations, which increase our confidence in both of them (since, as explained before, they use completely different approximations to the problem).

```
saedes_outputs.txt - Bloc de notas
Archivo  Edición  Buscar  Ayuda
SIMULATION OUTPUTS:
algorithm: SAEDES_A2
sim length: 6  seconds

target  availab.          99% CI
1:      0.955432    (0.954680 , 0.956184)
2:      0.813012    (0.811592 , 0.814432)
3:      0.673488    (0.671780 , 0.675196)
4:      0.579358    (0.577560 , 0.581156)
5:      0.528978    (0.527160 , 0.530796)
6:      0.504208    (0.502387 , 0.506029)
7:      0.495476    (0.493655 , 0.497297)
8:      0.494890    (0.493069 , 0.496711)
9:      0.498684    (0.496863 , 0.500505)
10:     0.501370    (0.499549 , 0.503191)
```

Figure 8: SAEDES++ Basic Outputs for System 1

Table 2: Comparison for Case 1 using 500,000 Iterations

Target time	SAEDES+	SAEDES++	Difference
1	0.955832	0.955432	0.0004
2	0.813418	0.813012	0.0004
3	0.673722	0.673488	0.0002
4	0.579418	0.579358	0.0001
5	0.527408	0.528978	-0.0016
6	0.503436	0.504208	-0.0008
7	0.495714	0.495476	0.0002
8	0.495990	0.494890	0.0011
9	0.497032	0.498684	-0.0017
10	0.501598	0.501370	0.0002
Simulation length	58 seconds	6 seconds	

As stated before, SAEDES++ is able to provide additional outputs. As can be seen in Figure 9, this system is expected to be available during 63% of the considered time interval (that is, between time $t_0 = 0$ and time $t_0 = 5$).

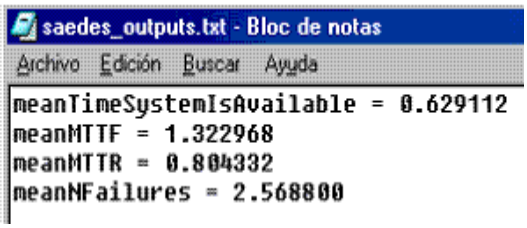

Figure 9: SAEDES++ Additional Outputs for System 1

9 STUDY CASE 2

A more complex system containing seven components, can be seen in Figure 10. Now, the system structure resembles similar to that of some telecommunication or computer network:

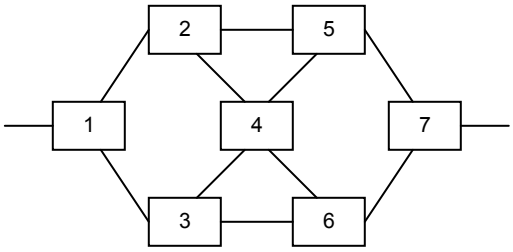

Figure 10: System for Case 2

Table 3 contains the failure-time distribution and repair-time distribution for each of the components. It is assumed, for simplicity reasons, that failure times and repair times follow exactly the same distribution. In contrast, Table 4 contains the minimal path decomposition of this system.

Table 3: Failure and Repair Times Distributions for Case 2

Component	Distribution	Parameter 1	Parameter 2
1	Weibull	1.8	2.8
2	Weibull	1.7	2.7
3	Weibull	1.6	2.6
4	Weibull	1.6	2.5
5	Weibull	1.4	2.4
6	Weibull	1.2	2.2
7	Weibull	1.3	2.3

Table 4: Minimal Path Structure for Case 2

Path	Composition
1	1 – 2 – 5 – 7
2	1 – 2 – 4 – 6 – 7
3	1 – 3 – 6 – 7
4	1 – 3 – 4 – 5 – 7

Again, we have used both SAEDES+ and SAEDES++ to obtain estimates for the system availability at ten different target times, $t_0 = 0.5,\ 1.0,\ 1.5,\ 2.0,\ ...,\ 5.0$.

Figure 11 shows results obtained with SAEDES+ (that is, using algorithm SAEDES_A1) after running 500,000 iterations:

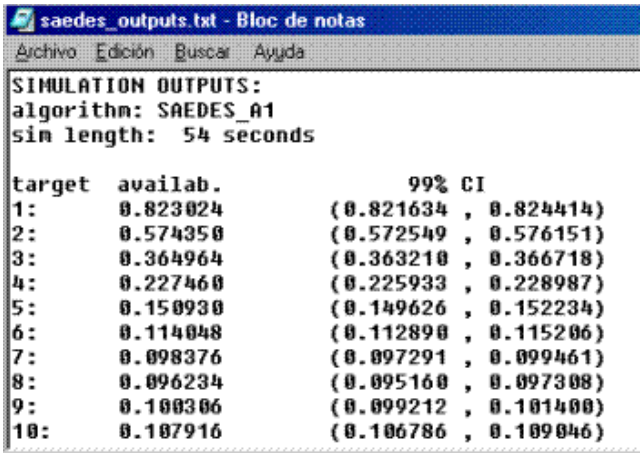

Figure 11: SAEDES+ Outputs for System 2

Once more, SAEDES+ has been able to provide us with point and interval estimates for the system availability at the ten different target times.

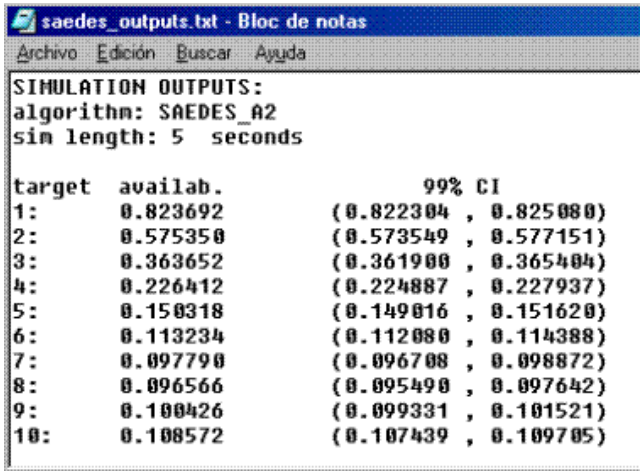

Figure 12: SAEDES++ Basic Outputs for System 2

The same system has been simulated with SAEDES++ (that is, using algorithm SAEDES_A2). Again, 500,000 iterations have been run. Figure 12 shows the basic outputs file.

Finally, Table 5 summarizes results from both programs and shows clearly that both algorithms provide, once more, convergent results.

Table 5: Comparison for Case 2 using 500,000 Iterations

Target time	SAEDES+	SAEDES++	Difference
1	0.823024	0.823692	-0.0007
2	0.574350	0.575350	-0.0010
3	0.364964	0.363652	0.0013
4	0.227460	0.226412	0.0010
5	0.150930	0.150318	0.0006
6	0.114048	0.113234	0.0008
7	0.098376	0.097790	0.0006
8	0.096234	0.096566	-0.0003
9	0.100306	0.100426	-0.0001
10	0.107916	0.108572	-0.0007
Sim length	54 seconds	5 seconds	

10 CONCLUSIONS

A simulation-based method called SAEDES has been presented in this paper. SAEDES can be very helpful for system managers and engineers in determining and improving complex systems availability. SAEDES is able to provide useful information about complex systems availability and can be applied in most situations where analytical methods are not well suited.

Two different and alternative algorithms have been developed to perform SAEDES core functions. Both algorithms have been implemented as computer programs and used separately to analyze different complex systems. Different case studies have been conducted, showing that results from both algorithms are convergent, which contributes to validate the method and to add credibility to it.

REFERENCES

Banks, J. (ed). 1998. *Simulation: principles, methodology, advances, applications, and practice*. New York: John Wiley & Sons.

Bajenescu, T. I. 1998. Predict the reliability of complex systems by applying the Monte Carlo method. In *Proceedings of the 6th International Conference on Optimization of Electrical and Electronic Equipments 1998. OPTIM '98,* 789-792. Brasov, Romania.

Barlow, R., and F. Proschan. 1996. *Mathematical theory of reliability*. Philadelphia. Society for Industrial & Applied Mathematics.

Billinton, R., and P. Wang. 1999. Teaching distribution systems reliability evaluation using Monte Carlo simulation. *IEEE Transactions on Power Systems* 14: 397-403.

Birolini, A. 2004. *Reliability engineering.* Springer-Verlag.

Collet, D. 2003. *Modeling survival data in medical research.* Boca Raton. Florida: Chapman & Hall/CRC.

Chisman, J. 1998. Using discrete simulation modeling to study large-scale system reliability/availability. *Computers and Operations Research* 25: 169-174.

Chung, C. 2004. Simulation modeling handbook: a practical approach. Boca Raton. Florida: CRC Press.

Goel, L., and R. Gupta. 1997. A windows-based simulation tool for reliability evaluation of electricity generating capacity. *International Journal of Engineering Education* 13: 347-357.

Hoyland, A., and M. Rausand. 1994. *System reliability theory: models and statistical methods.* New York: John Wiley-Interscience.

Juan, A., and A. Vila. 2002. SREMS: system reliability using Monte Carlo simulation with VBA and Excel. *Quality Engineering*, 15: 333-340.

Kamat, S., and M. Riley. 1975. Determination of reliability using event-based Monte Carlo simulation, *IEEE Transactions on Reliability* 24: 73-75.

Kovalenko, I.N., N.Y. Kuznetsov, and P.A. Pegg. 1997. *Mathematical theory of reliability of time dependent systems with practical applications*. New York: John Wiley & Sons, Inc.

Law, A., and D. Kelton. 2000. *Simulation modeling & analysis.* New York: McGraw-Hill.

Leemis, L. 2003. Input modeling. In *Proceedings of the Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J.S Carson, and M.S. Manivannan, pp. 62-73. Available via <http://www.informs-cs.org> [accessed January 4, 2005].

Levin, M., and T. Kalal. 2003. *Improving product reliability: strategies and implementation.* New York. John Wiley & Sons.

Lin, J., and C. Donaghey. 1993. A Monte Carlo simulation to determine minimal cut sets and system reliability. In *Proceedings Annual Reliability and Maintainability Symposium*, 246-249. Atlanta, GA.

L'Ecuyer, P. 2002. Random numbers. In the *International Encyclopedia of the Social and Behavioral Sciences*, ed. N.J. Smelser and P.B. Baltes, 12735-12738. Oxford.

Meeker, W., and L. Escobar. 1998. *Statistical methods for reliability data.* New York: John Wiley & Sons.

Pham, H. (ed) 2003. *Handbook of reliability engineering.* New York: Springer-Verlag.

Ross, S.M. 2001. *Simulation.* San Diego. California: Academic Press.

Shooman, M. 2002. *Reliability of computer systems and networks: fault tolerance, analysis, and design.* John Wiley & Sons.

Singh, C., and J. Mitra. 1995. Monte Carlo simulation for reliability analysis of emergency and standby power systems. In *Proceedings of the 30th IEEE Industry Applications Society Conference*, 2290-2295. Orlando, Florida.

Traub, R. 1994. *Reliability for the social sciences: theory and applications*. London: Sage Publications.

Wang, H., and H. Pham. 1997. Survey of reliability and availability evaluation of complex networks using monte carlo techniques. *Microelectronics Reliabiability* 37: 187-209.

Willis, R. 2000. Comparison of reliability-availability simulators. Available via <www.enre.umd.edu/rms/simultors.htm> [accessed February 4, 2005]

AUTHOR BIOGRAPHIES

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

ANGEL A. JUAN is an adjunct associate professor of Statistics at the Technical University of Catalonia (Barcelona, Spain). He is also an adjunct associate professor of Applied Mathematics at the Open University of Catalonia (UOC). His research interests include computer simulation and quantitative decision support systems. His e-mail address is <angel.alejandro.juan@upc.edu>.

VICENTE BARGUEÑO is an associate professor of Applied Mathematics at UNED (Madrid, Spain). His research interests include reliability problems and statistical modeling and analysis. He is also interested in distance teaching. His e-mail address is <vbargueno@ind.uned.es>.

CARLES SERRAT is an associate professor of Statistics at the Technical University of Catalonia (Barcelona, Spain. His research interests are related to Applied Statistics and Survival Analysis. His email address is <carles.serrat@upc.edu>.