Segregated Failures Model for Availability Evaluation of Fault-Tolerant Systems

Sergiy A. Vilkomir[1] **David L. Parnas**[1] **Veena B. Mendiratta**[2]
Eamonn Murphy[3]

[1]Software Quality Research Laboratory (SQRL), Department of Computer Science and Information Systems, University of Limerick, Limerick, Ireland,
Email: Sergiy.Vilkomir@ul.ie David.Parnas@ul.ie
[2]Bell Laboratories, Lucent Technologies, Naperville, IL, USA,
Email: Veena@lucent.com
[3]Department of Mathematics and Statistics, University of Limerick, Limerick, Ireland,
Email: Eamonn.Murphy@ul.ie

Abstract

This paper presents a method of estimating the availability of fault-tolerant computer systems with several recovery procedures. A segregated failures model has been proposed recently for this purpose. This paper provides further analysis and extension of this model. The segregated failures model is compared with a Markov chain model and is extended for the situation when the coverage factor is unknown and failure escalation rates must be used instead. This situation is illustrated in detail by estimating availability of a Lucent Technologies Reliable Clustered Computing architecture. For this example, numeric values are provided for availability indexes and the contribution of each recovery procedure to total system availability is analysed.
Keywords: software, fault-tolerance, availability, reliability, recovery, failures model.

1 Introduction

Some computer systems allow a variety of ways to restore service after a failure. We say such systems have several recovery procedures. For example, recovery procedure 1 can be a restart of a current application. If the restart of the application does not succeed, the computer can be restarted (recovery procedure 2). If the computer is still down after the restart, a repair or replacement is necessary (recovery procedure 3).

An example of an industrial computer system with several recovery procedures is a Lucent Technologies Reliable Clustered Computing (RCC) application (Hughes-Fenchel 1997). One of the basic recovery procedures for RCC is a switchover to a spare computer. Reliability and availability of systems with several recovery procedures have been studied by Hoeflin & Mendiratta (1995), Lyu & Mendiratta (1999), and Mendiratta (1998) for RCC products and by Ibe, Howe & Trivedi (1989), Sun, Han & Levendel (2001), and Sun, Han & Levendel (2003) for other systems.

Markov chains have been used as the most popular approach to reliability and availability investigation of systems with several recovery procedures (Ibe, Howe & Trivedi 1989, Lyu & Mendiratta 1999, Mendiratta 1998, Sun, Han & Levendel 2003). In Vilkomir *et al.* (2005), we have suggested a simpler analytical method of availability evaluation as an alternative. This method allows calculation of availability without using special tools and yields an assessment of the impact of every recovery procedure to system availability.

This paper continues the investigation started in Vilkomir *et al.* (2005) and considers the further analysis and extension of a segregated failures model. The paper is structured as follows. Section 2 presents a brief review of the segregated failures model based on Vilkomir *et al.* (2005) and then continues with new analysis and an extension of this model. In section 3, we compare our model with the Markov chain model. Small examples show the difference of approaches to availability evaluation for these two models and the advantages of using the segregated failures model are presented. We extend the model for the situation when coverage factors are unknown and show how to use rates of escalation instead. In section 4, a case study of an RCC application is considered in order to illustrate the extension of the segregated failures model. Because the RCC application has been previously considered by Lyu & Mendiratta (1999) using the Markov chain model, it provides an additional chance to compare the two approaches.

2 Segregated failures model of a system with several recovery procedures

Consider a system with n ($n > 1$) different recovery procedures. For every procedure from 1 to $n-1$, the result of the recovery can be either successful or unsuccessful. Level n recovery is always successful. When a failure occurs, recovery procedures are applied sequentially starting from level 1. If the recovery procedure at level 1 is unsuccessful, the level 2 procedure is applied, etc. However, if at any level it is determined for a specific failure that the usage of next recovery levels will not help, these levels can be skipped and the procedure of the last level n can be applied directly. Thus, there are three possibilities when a failure recovery is attempted at level $i, 1 \le i < n$:

- The recovery is successful.
- The recovery is unsuccessful and the next level procedure will be applied (the failure is *escalated* from level i to the next level $i+1$).
- The recovery is unsuccessful and the highest level procedure will be applied (the failure is escalated to level n).

These possibilities reflect real Lucent Technologies Reliable Clustered Computing applications as consid-

ered below in the Case Study section. The model can be extended to consider additional hypothetic possibilities such as:

- Skipping some restoration levels but not all of them.
- Using diagnostics that allows changing the order of recovery procedures for every specific failure depending on the nature of the failure.

We do not address these extensions in this paper but they are straightforward.

The ability of the recovery procedure to successfully restore a failure is described by a *coverage factor*. More precisely, a coverage factor $p_{rec,i}$ of the recovery level i is a conditional probability that a failure is successfully recovered at level i given that this failure is served at level i. In that way, a coverage factor $p_{rec,i}$ is the probability of the first of three mentioned above possibilities. Denote as $p_{next,i}$ the probability of second and as $p_{last,i}$ the probability of the third possibility.

When a recovery procedure is successful, we assume it provides full (not partial) recovery. This assumption is valid for many practical situations including the case study in section 4. Another assumption is that recovery duration is a random variable. This assumption is a traditional one and does not require special explanations. We consider the same distribution of recovery time whether the procedure is successful or not. To model it, we use restoration rate μ_i or mean restoration time $\tau_i = 1/\mu_i$. These indexes describe here only the duration of restoration, not its result (successful or unsuccessful). To highlight it, we will also use a term *mean processing time* for τ_i.

The main idea of the proposed approach is classifying processor failures into several types and evaluating the influence of each type of failure on the availability of the whole system. We propose the following definition of failure of *type i*: a failure f is said to be a failure of *type* i if and only if i is the lowest level where this failure is successfully recovered.

The described division of failures into types and the main parameters of the model are shown in Fig. 1, where λ is a system failure rate.

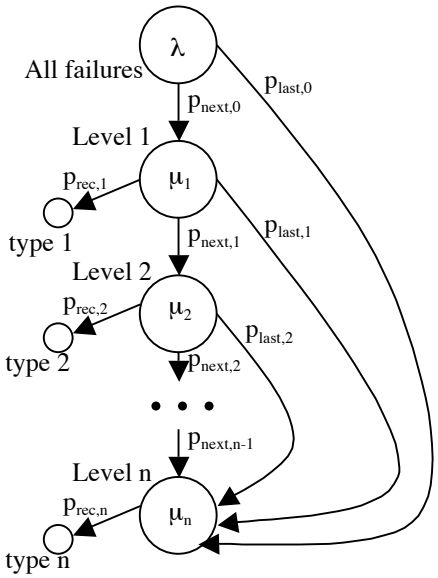

Figure 1: Segregated failures model

In Vilkomir *et al.* (2005), we had proposed using six main steps to evaluate availability according the segregated failures model.

At step 1, different types of failures should be determined corresponding to the recovery levels.

At step 2, probability p_{type_k} that a failure belongs type k is evaluated for each type k. For type 1,

$$p_{type_1} = p_{rec,1} \times p_{next,0} \tag{1}$$

For types k, $1 < k < n$,

$$p_{type_k} = p_{rec,k} \times \prod_{i=1}^{k-1} p_{next,i} \times p_{next,0} \tag{2}$$

For type n,

$$p_{type_n} = (p_{last,1} + \sum_{j=2}^{n-2}(p_{last,j} \times \prod_{i=1}^{j-1} p_{next,i}) + \\ + \prod_{i=1}^{n-1} p_{next,i}) \times p_{next,0} + p_{last,0} \tag{3}$$

At step 3, the failure rate λ_{type_k} is evaluated for failures of each type k:

$$\lambda_{type_k} = \lambda \times p_{type_k} \tag{4}$$

where p_{type_k} are determined by (1), (2), and (3) .

At step 4, the restoration rate μ_{type_k} is evaluated for failures of each type k. For failures of type k, the mean restoration time τ_{type_k} includes mean processing time τ_k at level k and also time which has been unsuccessfully spent on recovery at the previous levels:

$$\tau_{type_k} = \sum_{i=1}^{k} \tau_i \tag{5}$$

The restoration rate is:

$$\mu_{type_k} = \frac{1}{\sum_{i=1}^{k} \frac{1}{\mu_i}} \tag{6}$$

where μ_i is restoration (service) rate for the recovery procedure at level i.

At step 5, the availability is evaluated for failures of each type k. The expected down time T_{dk} during a fixed period of time T relative to failures of type k is:

$$T_{dk}(T) = T(1 - A_k) = T\frac{\lambda_{type_k}}{\lambda_{type_k} + \mu_{type_k}} \tag{7}$$

where $A_k = \mu_{type_k}/(\lambda_{type_k} + \mu_{type_k})$ - availability factor.

When $\lambda_{type_k} \ll \mu_{type_k}$, it is possible to use the approximate value of the down time per year:

$$T_{dk} = \lambda_{type_k}\tau_{type_k} \tag{8}$$

calculating λ_{type_k} in 'failures per year'and τ_{type_k} in minutes.

At step 6, the down time of the whole system is evaluated:

$$T_d = \sum_{k=1}^{n} T_{dk} \tag{9}$$

3 Analysis and further extension of the model

3.1 Comparison with a Markov chain model

In this section we illustrate similarities and differences between the Markov chain model and the segregated failures model. The Markov chain model for a system with several recovery levels is illustrated in Fig. 2. Despite a certain similarity between Fig. 1 (Segregated failures model) and Fig. 2 (Markov chain model), they have different meanings.

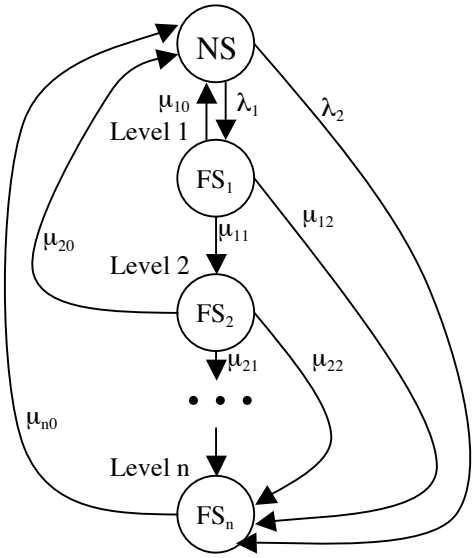

Figure 2: Markov chain model.

The circles in Fig. 2 represent states of a system: one working state and several faulted states for every recovery level. The arrows represent transitions between system states. In contrast, the circles in Fig. 1 represent sets of failures. The arrows represent relationships between these sets, i.e., how one set of failures is divided into other sets (with corresponding probabilities). The two models use the same input data and lead to the same results but use different approaches to system availability evaluation.

The Markov chain model is a powerful mathematical approach and allows modelling of many aspects of a system's behaviour, not just availability. Using the Markov chain model, the probabilities of system states are evaluated. Knowing the probability of a normal (working) state, the system availability can be evaluated. However, calculations based on this model can be quite complicated (solving the Chapman-Kolmogorov equations). Special tools are often required for this type of analysis. The segregated failures model is designed for a specific purpose - availability evaluation of systems with several recovery procedures. System states are not considered and an impact of different types of failures on system availability is considered instead. In contrast to the Markov chain model, calculations are very simple and do not required of using any tools.

The segregated failures model is proposed not instead of but in addition to the Markov chain model. Taking into account the computational complexity of the Markov chain model, we believe it is useful to have a simple engineering analytical method of availability evaluation. To illustrate this, consider an application of both models to the following simple toy example:

- A system has two different recovery procedures.
- The probability that a failure is recovered by the first procedure is 2/3.
- The mean restoration time for the first recovery procedure is 30 times less then for the second one.

Both models for this case are represented in Fig. 3, where

- λ - system failure rate.
- μ - recovery rate for the second procedure.
- $P_i, i = 0, 1, 2$ - probabilities of system states.

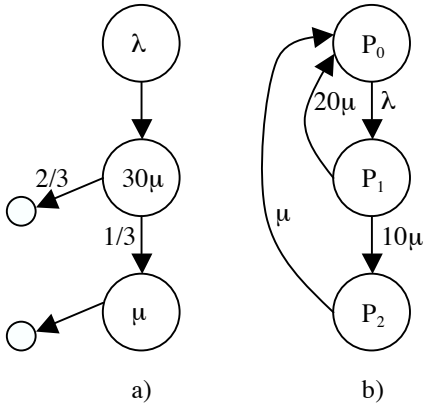

Figure 3: System with two recovery procedures: a) Segregated failures model and b) Markov chain model

Application of the segregated failures model for this example is so simple that it requires only mental calculations. Thus, the failure rates for failures of the each type from (4) are

$$\lambda_{type_1} = \frac{2}{3}\lambda; \quad \lambda_{type_2} = \frac{1}{3}\lambda \tag{10}$$

The mean restoration times for failures of the each type from (5) are

$$\tau_{type_1} = \frac{1}{30\mu}; \quad \tau_{type_2} = \frac{1}{30\mu} + \frac{1}{\mu} = \frac{31}{30\mu} \tag{11}$$

Finally, the system down time can be found using (9):

$$T_d = \frac{2}{3}\lambda\frac{1}{30\mu} + \frac{1}{3}\lambda\frac{31}{30\mu} = \frac{11\lambda}{30\mu} \tag{12}$$

Application of the Markov chain model is slightly more complicated. We need to solve the following simultaneous equations:

$$\lambda P_0 = 20\mu P_1 + \mu P_2 \tag{13}$$

$$30\mu P_1 = \lambda P_0 \tag{14}$$

$$\mu P_2 = 10\mu P_1 \tag{15}$$

$$P_0 + P_1 + P_2 = 1 \tag{16}$$

Transposing (14) for P_1 and (15) for P_2 and substituting P_1 and P_2 into (16) gives

$$P_0 + \frac{\lambda}{30\mu}P_0 + \frac{\lambda}{3\mu}P_0 = 1 \tag{17}$$

and

$$P_0 = \frac{30\mu}{11\lambda + 30\mu} \tag{18}$$

The system down time during time period T can be expressed as

$$T_d = (1 - P_0)T \tag{19}$$

Considering down time during $T = 1year$ and using (18) , we finally have

$$T_d = \frac{11\lambda}{11\lambda + 30\mu} \tag{20}$$

In practice usually $\lambda_{type_k} \ll \mu_{type_k}$ and from (20) the approximate value of the down time is

$$T_d = \frac{11\lambda}{30\mu} \tag{21}$$

which completely coincides with result (12) of the segregated failures model.

The following conclusions can be drawn from the example:

- The segregated failures model provides the approximate values of the down time which are very close to the accurate values. Thus, if $\lambda = 10$ *per year* and $\mu = 1$ *per hour*, the accurate value of T_d according (20) is 219.91 *min per year*. The approximate value of T_d according to (12) or (21) is 220.00 *min per year*. The difference is only 0.04% and is negligible, especially taking into account an approximation of the input data.
- The complexity of calculations according to the Markov chain model increases when the number of recovery procedures increases. At the same time, the complexity of calculations according to the segregated failures model changes insignificantly. Thus, the benefit of using the segregated failures model increases when more recovery procedure are used.
- Both the Markov chain model and the segregated failures model allow us to evaluate the system down time. However, the segregated failures model also provides a separate evaluation of the down times for each recovery procedure. This in turn allows us to analyze availability in more detail and to find ways to improve availability.

3.2 Rates of escalation instead of the coverage factor

In this section we propose an extension of the segregated failures model for the situation when the input data are different from what was considered previously. As was mentioned in Section 2, we assumed that some conditional (given that level i procedure is applied) probabilities are known. Specifically, probability that a failure recovery is successful ($p_{rec,i}$) or that a failure recovery is unsuccessful and that the next recovery level is either level $i + 1$ ($p_{next,i}$) or last level n ($p_{last,i}$). For applications of the Markov chain model, explicit rates of transitions between system states are often used as input data instead of these probabilities. This situation is also possible for the segregated failures model. In this case, the input data for the model are:

- $\mu_{rec,i}$ - successful failure recovery rate.
- $\mu_{next,i}$ - rate of the escalation from level i to the next level $i + 1$.
- $\mu_{last,i}$ - rate of the escalation from level i to the last level n.

These three possibilities are mutually exclusive and exhaustive. So the failure *processing* rate μ_i at level i (i.e. failure exit rate regardless of the results of recovery) is a sum of rates for these possibilities:

$$\mu_i = \mu_{rec,i} + \mu_{next,i} + \mu_{last,i} \tag{22}$$

The best way to evaluate availability in this situation is to express $p_{rec,i}$, $p_{next,i}$, and $p_{last,i}$ using $\mu_{rec,i}$, $\mu_{next,i}$ and $\mu_{last,i}$ and then to apply the basic segregated failures model described in Section 2. Each probability is determined as a ratio of the corresponding rate to the failure processing rate of the whole procedure:

$$p_{rec,i} = \frac{\mu_{rec,i}}{\mu_i} = \frac{\mu_{rec,i}}{\mu_{rec,i} + \mu_{next,i} + \mu_{last,i}} \tag{23}$$

$$p_{next,i} = \frac{\mu_{next,i}}{\mu_i} = \frac{\mu_{next,i}}{\mu_{rec,i} + \mu_{next,i} + \mu_{last,i}} \tag{24}$$

$$p_{last,i} = \frac{\mu_{last,i}}{\mu_i} = \frac{\mu_{last,i}}{\mu_{rec,i} + \mu_{next,i} + \mu_{last,i}} \tag{25}$$

To apply the model from Section 2, we also need to express mean *service* (processing) time τ_i at level i. For traditional systems with one recovery procedure, the mean restoration time is a reciprocal value of the failure restoration rate. For systems with several recovery procedures, there are different rates for each level, in particular, rates of successful failure recovery $\mu_{rec,i}$ and failure processing rate μ_i. Because we assume that the mean processing time for a specific level is the same for all failures and independent of restoration results, this time is a reciprocal value of the failure processing rate, not of the successful recovery rate. In other words,

$$\tau_i = \frac{1}{\mu_i} = \frac{1}{\mu_{rec,i} + \mu_{next,i} + \mu_{last,i}} \tag{26}$$

Using (23) - (26) allows us to evaluate availability of a system with known explicit rates of escalation, leading to the situation described in Section 2.

4 A Case Study: Describing recovery policy by rates of escalation without the use of coverage factor

4.1 A model

As an example of a model with rates of escalation we consider a hypothetical RCC application, which has been analyzed by Lyu & Mendiratta (1999) using a Markov chain model. We reuse notation and inputs from Lyu & Mendiratta (1999) in our model which allows us to compare different approaches to availability evaluation.

The model has four recovery procedures:

- Fault Detection and Recovery. A small number of hardware and software faults are detected by the watchdog and recovery is fully automatic. The internal data are not saved and the application is restarted at the initial internal state.
- Volatile Data Recovery. Periodic checkpointing is used and the critical volatile data are saved. The process automatically restarts at the most recent checkpointed internal state.

- Persistent Data Recovery. Replication of the persistent data on a backup disk is carried out. This ensures data consistency when the application is automatically recovered on the backup node.
- Manual Repair. For all hardware and software faults when attempts of automatic recovery are not successful, manual intervention is used. In addition, a small set of faults is detected for manual repair before applying procedures of the automatic recovery.

A diagrammatic representation of the model is shown in Fig. 4.

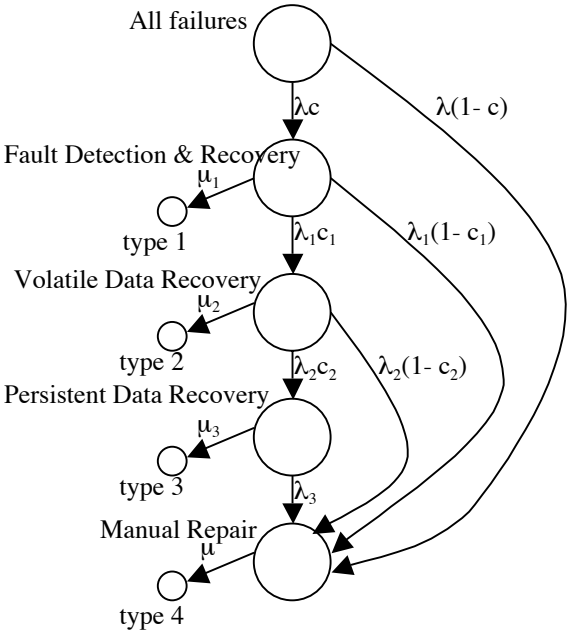

Figure 4: Segregated failures model with rates of escalation.

Model inputs are the following:

- λ - total failure rate.
- $\lambda_i, i = 1, 2, 3$ - level i to $i + 1$ escalation rate.
- $\mu_i, i = 1, 2, 3$ - recovery rate at level i.
- μ - manual repair rate.
- c - fault detection coverage.
- $c_i, i = 1, 2$ - level i to $i + 1$ coverage.

The values of input data are the following (Lyu & Mendiratta 1999):
$\lambda = 10, 20, 30$ failures per year
$\lambda_1 = 30$ exits per hour
$\lambda_2 = 1800$ exits per hour
$\lambda_3 = 100$ exits per hour
$\mu_1 = 30, 60$ recoveries per hour
$\mu_2 = 1800, 3600$ recoveries per hour
$\mu_3 = 3600$ recoveries per hour
$\mu = 0.25$ repair per hour
$c, c_1, c_2 = 0.9, 0.99$

4.2 Availability evaluation

Step 1: The model has the following four failure types:

- Type 1: failures restored by the fault detection and recovery procedure.
- Type 2: failures restored by the volatile data recovery procedure.
- Type 3: failures restored by the persistent data recovery procedure.
- Type 4: failures restored by the manual repair procedure.

Step 2: To turn from rates of escalation to coverage factors, let us calculate conditional probabilities $p_{rec,i}$, $p_{next,i}$, and $p_{last,i}$. The application of (23) - (25) gives the following:

$$p_{rec,i} = \frac{\mu_i}{\mu_i + \lambda_i}, i = 1, 2, 3 \quad (27)$$

$$p_{next,i} = \frac{\lambda_i c_i}{\mu_i + \lambda_i}, i = 1, 2 \quad (28)$$

$$p_{next,3} = \frac{\lambda_3}{\mu_3 + \lambda_3} \quad (29)$$

$$p_{last,i} = \frac{\lambda_i (1 - c_i)}{\mu_i + \lambda_i}, i = 1, 2 \quad (30)$$

Now we can calculate p_{type_i} applying (1) - (3):

$$p_{type_1} = \frac{c\mu_1}{\mu_1 + \lambda_1} \quad (31)$$

$$p_{type_2} = \frac{c\lambda_1 c_1 \mu_2}{(\mu_1 + \lambda_1)(\mu_2 + \lambda_2)} \quad (32)$$

$$p_{type_3} = \frac{c\lambda_1 c_1 \lambda_2 c_2 \mu_3}{(\mu_1 + \lambda_1)(\mu_2 + \lambda_2)(\mu_3 + \lambda_3)} \quad (33)$$

$$p_{type_4} = c \times \left(\frac{\lambda_1(1 - c_1)}{(\mu_1 + \lambda_1)} + \frac{\lambda_1 c_1 \lambda_2 (1 - c_2)}{(\mu_1 + \lambda_1)(\mu_2 + \lambda_2)} + \right.$$
$$\left. + \frac{\lambda_1 c_1 \lambda_2 c_2 \lambda_3}{(\mu_1 + \lambda_1)(\mu_2 + \lambda_2)(\mu_3 + \lambda_3)}\right) + 1 - c \quad (34)$$

Step 3: use (4) and values of $p_{rec,i}$, $p_{next,i}$, and $p_{last,i}$ (obtained at Step 2) for the calculation of λ_{type_i}.

Step 4: For the evaluation of the mean restoration time τ_{type_i} for failures of the every type i, we firstly calculate the mean processing time τ_i for the every level i using formulas $\tau_i = \frac{1}{\mu_i + \lambda_i}, i = 1, 2, 3$ and $\tau_4 = \frac{1}{\mu_4}$. Then we find τ_{type_i} using (5). The results of the calculation are shown in Table 1.

Level/ Type	Mean restoration time	$\mu_1 = 30$ $\mu_2 = 1800$	$\mu_1 = 60$ $\mu_2 = 3600$
1	τ_1	1.0	0.67
	τ_{type_1}	1.0	0.67
2	τ_2	0.02	0.01
	τ_{type_2}	1.02	0.68
3	τ_3	0.02	0.02
	τ_{type_3}	1.03	0.7
4	τ_4	240	240
	τ_{type_4}	241.03	240.7

Table 1: Mean restoration time (minutes).

Step 5 - 6: The intermediate results and values T_{di} and T_d of the down time according to (8) - (9) are presented in Table 2 for $\mu_1 = 30, \mu_2 = 1800$ and Table 3 for $\mu_1 = 60, \mu_2 = 3600$.

The comparison of these results with the results from Lyu & Mendiratta (1999), where the same case study has been analyzed using the Markov chain model, shows that the values of the down time from

Type of failures	Failure rate and down time	$c_i = 0.99$			$c_i = 0.9$		
		$\lambda = 10$	$\lambda = 20$	$\lambda = 30$	$\lambda = 10$	$\lambda = 20$	$\lambda = 30$
1	λ_{type_1}	4.95	9.9	14.85	4.50	9.00	13.50
	T_{d1}	5.0	9.9	14.9	4.5	9.0	13.5
2	λ_{type_2}	2.45	4.9	7.35	2.03	4.05	6.07
	T_{d2}	2.5	5.0	7.5	2.1	4.1	6.2
3	λ_{type_3}	2.36	4.72	7.08	1.77	3.55	5.32
	T_{d3}	2.4	4.9	7.3	1.8	3.7	5.5
4	λ_{type_4}	0.24	0.48	0.72	1.7	3.40	5.11
	T_{d4}	57.9	115.7	173.6	410.2	820.5	1230.7
System down time T_d		68	135	203	419	837	1256

Table 2: Failure rates (per year) and expected down time (minutes per year) for $\mu_1 = 30, \mu_2 = 1800$.

Type of failures	Failure rate and down time	$c_i = 0.99$			$c_i = 0.9$		
		$\lambda = 10$	$\lambda = 20$	$\lambda = 30$	$\lambda = 10$	$\lambda = 20$	$\lambda = 305$
1	λ_{type_1}	6.60	13.20	19.80	6.00	12.00	18.00
	T_{d1}	4.4	8.8	13.3	4.0	8.0	12.1
2	λ_{type_2}	2.18	4.36	6.53	1.80	3.60	5.40
	T_{d2}	1.5	2.9	4.4	1.2	2.4	3.7
3	λ_{type_3}	1.05	2.10	3.15	0.79	1.58	2.36
	T_{d3}	0.7	1.5	2.2	0.6	1.1	1.7
4	λ_{type_4}	0.17	0.34	0.52	1.41	2.82	4.24
	T_{d4}	41.6	83.3	124.9	339.9	679.7	1019.6
System down time T_d		48	97	145	346	691	1037

Table 3: Failure rates (per year) and expected down time (minutes per year) for $\mu_1 = 60, \mu_2 = 3600$.

both models are practically the same. The difference is on average less than 0.5%, which can be explained by rounding off the decimal. Whereas results from Lyu & Mendiratta (1999) provide only the system down time values, Tables 2 and 3 also provide the down time for each recovery procedure.

Two conclusions can be directly drawn from Tables 2 and 3:

- System down time is proportional to the value of the total failure rate; that is also clear from the formulas (4) and (8).
- Increasing the value of the fault detection coverage c_i significantly decreases down time.

The value of the fault detection coverage c_i determines the number of failures that are escalated from the level i to the next level (not to last level directly). Increasing c_i from 0.9 to 0.99 decreases down time several times (6.2 times from 419 min to 68 min for $\mu_1 = 30, \mu_2 = 1800$; 7.2 times from 346 min to 48 min for $\mu_1 = 60, \mu_2 = 3600$). However, the fault detection coverage value depends on the nature of software failures and cannot always be changed by system designers. Furthermore, systems with a bad diagnostic subsystem (which cannot determine failures requiring immediate manual repair) have a greater fault detection coverage value. But if some failures are eventually escalated (step by step) through all levels to the last one, system down time will increase because of the time lost at each level.

There are at least two ways for designers to improve availability of a system:

- Change recovery strategy.
- Improve individual recovery procedures.

Changing the existing recovery strategy by creating new recovery procedures can require significant effort from designers. However, in some cases it is possible to improve availability just by changing the order in which recovery procedures are applied. The segregated failures model can be useful for studying such changing.

An existing recovery procedure can be improved by reducing mean recovery time for the procedure. High recovery time for a specific procedure influences system availability even when the number of failures restored at this level is negligible. Thus, according to Tables 2 and 3 for $c_i = 0.99$, on average only 2.1% of all failures are restored at level 4, i.e. by manual repair. However, the contribution of this level to system down time comes to 86% (85.1% for $\mu_1 = 30, \mu_2 = 1800$ and 86.7% for $\mu_1 = 60, \mu_2 = 3600$). The reason is that, according to Table 1, the mean restoration time for the manual repair is two hundred times more than the mean restoration time for other procedures.

Mean restoration time can be reduced by using better diagnostic equipment, speeding up a delivery of spare parts, etc. Thus, if the mean restoration time for the manual repair is reduced by 20% (from 240 min to 192 min) then, according to (8) and (9), system down time will be reduced on average by 18%.

5 Conclusion

In this paper, we considered fault-tolerant computer systems with several recovery procedures. We continued the earlier investigations described in Vilkomir *et al.* (2005) by investigating the segregated failures model for availability evaluation of such systems. This model provides a simple analytical method of evaluating system availability and can be used as an alternative to Markov chain models. We analysed both approaches and showed that the segregated failures model not only allows us to calculate down time of a whole system but also gives the possibility, using intermediate results of calculations, of evaluating the impact of each recovery procedure on system availability. The model can also be used for the correction of a recovery strategy and improvement of system availability.

We extended the segregated failures model for the situation when the values of implicit rates of failures escalation are known instead of coverage factors. As

a case study, a Lucent Technologies Reliable Clustered Computing architecture was considered. Different values of failure restoration rates were considered and their influence on down time was analysed. The values of down time for every type of failures as well as for the whole system were calculated. For this specific case study, the main factor that impacted system availability was the mean restoration time for the manual repair. Thus, reducing this time is an important practical task to improve availability.

The case study shows that the segregated failures model provides a simple, convenient and practical approach for availability evaluation. In future work, we will consider an application of this model to the analysis of different recovery strategies and selection of an optimal strategy for any given system.

6 Acknowledgement

This work was supported by Science Foundation Ireland under SFI Grants 01/P1.2/C009 and 03/CE3/1405.

References

Hoeflin, D.A. & Mendiratta, V.B. (1995), Elementary Model for Predicting Switching System Outage Durations, *in* 'Proceedings of XV International Switching Symposium', Berlin, Germany, 23–28 April, 1995.

Hughes-Fenchel, G. (1997), A flexible clustered approach to high availability, *in* 'Digest of Papers of Twenty-Seventh Annual International Symposium on Fault-Tolerant Computing', FTCS-27, Seattle, Washington, USA, 24–27 June 1997, pp. 314–318.

Ibe O., Howe, R. & Trivedi, K. S. (1989), Approximate Availability Analysis of VAXCluster Systems, *IEEE Transactions on Reliability*, Vol. 38, No. 1, April 1989, pp. 146–152.

Lyu, M.R. & Mendiratta, V.B. (1999), Software fault tolerance in a clustered architecture: techniques and reliability modelling, *in* 'Proceedings of the 1999 IEEE Aerospace Conference', Volume 5, Snowmass, CO, USA, 6–13 March 1999, pp. 141–150.

Mendiratta, V.B. (1998), Reliability analysis of clustered computing systems, *in* 'Proceedings of the Ninth International Symposium on Software Reliability Engineering', Paderborn, Germany, 4–7 November 1998, pp. 268–272.

Sun, H., Han, J.J. & Levendel, H. (2001), A generic availability model for clustered computing systems, *in* 'Proceedings of 2001 Pacific Rim International Symposium on Dependable Computing', Seoul, Korea, 17–19 December 2001, pp. 241–248.

Sun, H., Han, J.J. & Levendel, H. (2003), Availability requirement for a fault-management server in high-availability communication systems, *IEEE Transactions on Reliability*, Volume 52, Issue 2, June 2003, pp. 238–244.

Vilkomir S., Parnas D., Mendiratta V. & Murphy E. (2005), Availability evaluation of hardware/software systems with several recovery procedures, *in* 'Proceedings of the 29th IEEE Annual International Computer Software and Applications Conference' (COMPSAC 2005), IEEE Computer Society, Volume 1, Edinburgh, Scotland, 25–28 July 2005, pp. 473–478.