# A PRACTICAL DEPENDABILITY MEASURE FOR DEGRADABLE COMPUTER SYSTEMS WITH NON–EXPONENTIAL DEGRADATION

E. JONSSON [1], M. ANDERSSON [2], S. ASMUSSEN [3]

[1] *Department of Computer Engineering, Chalmers University of Technology, Sweden*
[2] *Department of Mathematics, Chalmers University of Technology, Sweden*
[3] *Institute of Electronic Systems, Aalborg University, Denmark*

**Abstract.** This paper proposes a vectorized measure for a set of behavioural dependability attributes. The measure is based on Markov processes and is intended for practical dependability trade-offs. It describes the system performance on a number of *service levels*. Thus, it is possible to merge attributes such as *reliability*, *safety* and *performability* into one single quality. Whereas reliability describes the functional fulfillment of a system, performability reflects its ability of functional degradation. The safety attribute handles a class of failures with catastrophic consequences and can be accommodated by means of introducing two or more service levels for a failed system. Many systems exhibit time-dependent degradation rates and non–exponential lifetimes. This situation can be handled by means of applying phase–type assumptions and introducing some additional states to the system, which would allow us to remain within the universe of Markov modelling.

**Keywords.** Dependability, Safety, Performability, Measure, Computer System

## 1. INTRODUCTION

The classical viewpoint of dependability is given in (Laprie 1992). Dependability is described by four basic attributes which are primarily related to non-degradable systems: *reliability, availability, safety* and *security*. Furthermore, *performability* is a performance-related attribute used for degradable systems, see e.g. (Beaudry 1978, Meyer 1980, Smith and Trivedi 1987) and the comprehensive overview in (Meyer 1992). This paper suggests a measure for a set of *behavioural* dependability attributes as defined in section 2. The measure is defined in section 3, and two different methods for computation of the measure are given in section 4. The measure does not reflect aspects of self-repair or automatic up-grading, nor does it include external maintenance or repair. Thus, the availability attribute is not covered by the measure. The rationale for this is that measures of availability are usually based on a binary notion of the system's performance and are therefore unable to convey intermediate levels of readiness. There is no widely recognized or obvious definition that would account for availability reduction due to degraded operational states (Meyer 1992).

In many cases the assumption of exponential failure rates for the components of the system is not realistic. This situation can be handled using phase–type assumptions as described in section 5.

## 2. BEHAVIOURAL ATTRIBUTES

The measure derived in this paper is based on the traditional dependability attributes as given in the introduction, including performability. However, a behavioural view of the attributes is taken. Here, behavioural refers to the behaviour of the *object system* as seen by its *user* and related to the *service* delivered from the system. Thus, the measure reflects reliability for operational service levels and safety for non-operational levels.

*Security* is covered to the extent that it describes behavioural aspects, i.e. privacy (or confidentiality) considerations. Whereas *reliability* refers to the *delivery-of-service* to the *authorized user* of the system, *privacy* refers to the *denial-of-service* to *unauthorized users.* See (Jonsson and Olovsson 1992). This means that unauthorized users shall not be able to get information from the system, nor be able to use it in any other way. Degradations or failures can be due to "reliability" faults as well as "privacy" faults. Note that safety is the non-occurrence of catastrophic failures of both those kinds. Privacy aspects are not further considered in this paper. A treatment of these is found in (Jonsson *et al.* 1994).

Another aspect of security is its ability to resist external faults, in particular intentional faults. This aspect relates to fault prevention or intrusion-tolerance and is not directly covered by the measure described in this paper.

## 3. A DEPENDABILITY MEASURE

This section provides a mathematical definition of the vectorized dependability measure, as well as presents the method and equations used for the calculations. The method is based on a predefined set of service levels and a set of corresponding failure rates, which quantify the rate of transitions between levels. A **service level** is defined as a group of system states, each with a user-specified degree of performance or functional accomplishment.

We shall assume that the state of the system can be modelled as a continuous time Markov process $\{X_t\}_{t\geq 0}$ with a finite state space $E$, in which each service level, $SLn$, can be identified with a subset of states in $E$. Thus, $E$ is the disjoint union $SL0 + \cdots + SL\ell$, where $\ell$ is the number of service levels. Further, we imagine that service levels $0, \ldots, k$ correspond to operational states $O$, i.e. the states in which the system functions, in the sense that it delivers a full, $SL0$, or degraded service to the user. Thus, the system *reliability* can be modelled.

Service levels $k+1, \ldots, \ell$ correspond to the *failed* states $F$, i.e. states in which the system is not functioning, meaning that it is not delivering any service of interest to the user. The reason for introducing service levels for failed states is to model the severity of the consequences of the failure involved, which is especially relevant for safety-critical systems, rather than the service delivered to the user as in operational states. In the simplest case we group failures into two classes, non-catastrophic and catastrophic. The ability of a system to avoid catastrophic failures is the system *safety.* Thus

$$\begin{aligned} E &= O + F \qquad \text{where} \\ O &= SL0 + \cdots + SLk, \\ F &= SL(k+1) + \cdots + SL\ell. \end{aligned}$$

In the simplest case, corresponding to the traditional operational-failed model, $O$ consists of one single state $o$, and $F$ consists of one single state $f$. In more complex situations, the different states in $O$ represent different full or degraded service levels, and $F$ represents different types of failed states.

Transitions $i \to j$ have intensity $\lambda_{ij}$ $(i, j \in E$, $i \neq j)$, and the initial probability $\mathbb{P}(X_0 = i)$ is denoted by $\pi_i$. In most situations, the system will always start in a fixed state $i_0$ so that

$$\pi_j = \begin{cases} 1 & j = i_0 \\ 0 & j \neq i_0 \end{cases} . \tag{1}$$

We shall also assume that the system starts at the highest service level, so that $i_0 \in SL0$. Transitions between operational states represent *degradations*, and transitions to a failed state represent *failures.* It is assumed that a transition will never take place from a failed state, i.e. after entering a failed state the system stops evolving. Therefore, failed states are absorbing so that $\lambda_{fj} = 0$ for $f \in F$ and all $j \in E$. For mathematical convenience we shall denote the intensity for leaving state $i$ as $\lambda_i = \sum_{j\neq i} \lambda_{ij}$, and we write $\lambda_{ii} = -\lambda_i$.

Assuming that $O$ has $n$ states and $F$ $m$, we suggest the $n+m$ vector

$$\mathbf{w} \;=\; \left((u_i)_{i\in O}, (v_i)_{i\in F}\right), \tag{2}$$

as a measure of dependability of the system (the "dependability" vector). Here $u_i$ is the mean time in state $i$ or **Mean Time To Degradation** (= MTTD) and $v_i$ is the Mean Time To Failure (=MTTF), i.e. the sum of the MTTDs of the operational states, divided by the probability $p_i$ of ending up in the failed state $i$. The measures $v_i$ that we allocate to the failed states represent a splitting of the mean *operational* lifetime of a sequence of identical systems, so that a more probable failed state receives a smaller allocation than a less probable state. The lowest state normally represents a catastrophic failure. Obviously we want the value for a catastrophic state to be as large as possible. In formal mathematical terms we have

$$u_i \;=\; \mathbb{E}\int_0^\infty I(X_t = i)dt, \quad i \in O, \tag{3}$$

$$v_i \;=\; \frac{1}{p_i}\sum_{j\in O} u_j, \quad i \in F, \tag{4}$$

where $I$ is the indicator function (i.e., $I(X_t = i) = 1$ in (3) if $X_t = i$ and $I(X_t = i) = 0$ if $X_t \neq i$). For computational purposes, we note that $u_i$ and the probability $p_i$ that the system ever enters $i$ can be denoted

$$u_i = \frac{p_i}{\lambda_i}, \quad p_i = \mathbb{P}(\tau_i < \infty), \quad i \in E = O + F, \quad (5)$$

since we have assumed that there is no feedback. Here $\tau_i = \inf\{t \geq 0 : X_t = i\}$ ($\tau_i = \infty$ if no $t$ with $X_t = i$ exists) is the hitting time of $i$, i.e. the time of first entry. In order to illustrate the use of the dependability measure (2) a very simple example is given below.

Example 1: Consider a process control system $P$, consisting of three units: a high-performance, real-time computer $C$ for basic process support, i.e. "survival" functions; a general purpose computer $B$ for direct process control, i.e. controlling the characteristics of the product to be produced; and one I/O computer $A$ for remote control, hard-copy print-outs etc. If $C$ fails, the process will be interrupted and a re-start has to be initiated at considerable cost. This is considered a catastrophic failure. If $B$ fails the process as such will continue, but there will be no meaningful product. This is regarded as a "normal" system failure. Finally, a disruption of the function of $A$ means that remote control is disabled and that logged info cannot be printed out immediately. However, as the process can still be controlled locally, this is only interpreted as a degradation.

In view of the discussion above, the service levels may be interpreted as

$$\begin{aligned} SL0 &= ABC \quad \text{(full service level)} \\ SL1 &= aBC \quad \text{(degraded service level)} \\ SL2 &= AbC + abC \quad \text{(failed service level)} \\ SL3 &= **c \quad \text{(catastrophically failed service level)} \end{aligned}$$

where an upper–case letter, e.g. $A$, denotes a component that is functioning and a lower–case letter, e.g. $a$, that it has failed. We can identify $E$ with $\{SL0, SL1, SL2, SL3\}$ where $O = \{SL0, SL1\}$, $F = \{SL2, SL3\}$. A state diagram for the system is given in Fig. 1.

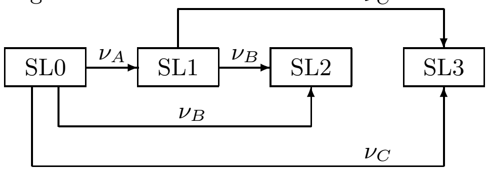

Figure 1. State diagram for process system

Here $\nu_A$ denotes the failure intensity for component $A$, etc. We see that

$$\begin{aligned} u_{SL0} &= \frac{1}{\nu_A + \nu_B + \nu_C}, \\ p_{SL1} &= \frac{\nu_A}{\nu_A + \nu_B + \nu_C}, \quad u_{SL1} = p_{SL1} \cdot \frac{1}{\nu_B + \nu_C}, \\ p_{SL2} &= p_{SL1} \cdot \frac{\nu_B}{\nu_B + \nu_C} + \frac{\nu_B}{\nu_A + \nu_B + \nu_C} = \frac{\nu_B}{\nu_B + \nu_C}, \\ p_{SL3} &= p_{SL1} \cdot \frac{\nu_C}{\nu_B + \nu_C} + \frac{\nu_C}{\nu_A + \nu_B + \nu_C} = \frac{\nu_C}{\nu_B + \nu_C}, \\ v_{SL2} &= \frac{u_{SL0} + u_{SL1}}{p_{SL2}} = \cdots = \frac{1}{\nu_B}, \\ v_{SL3} &= \frac{u_{SL0} + u_{SL1}}{p_{SL3}} = \cdots = \frac{1}{\nu_C}, \end{aligned}$$

In this simple example we see that the entries for the failed states are simply the inverse of the transition rate to that state. Inserting numerical values $\nu_A = \nu_B = 9.5/10000$ failures per hour and $\nu_C = 1/10000$ failures per hour, yields the dependability vector

$$\mathbf{w} = (500 \;\; 452 \;\; 1053 \;\; 10000)\,, \quad (6)$$

where the higher figure on the lowest level represents the fact that a catastrophic failure is much less probable than is the other type of failure. □

## 4. COMPUTATION PROCEDURES

In this section, we develop two different algorithms for computing the dependability measure (2). The first algorithm is a hierarchical procedure, which in essence was used in the first example, even though the formalism was not made evident there. Here, the word hierarchical is used in the sense "no feed–back", i.e. the state diagram contains no loops. Thus, the hierarchical approach is suitable for calculations by hand for small systems.

The second algorithm is based on a general matrix formalism that may well be used for larger systems, when computer-aided tools are used. The matrix approach can be described by a compact mathematical form and it is applicable also in the absence of a hierarchical structure (i.e. feed–back is possible). Only when systems become so large that the matrix inversion presents a problem, does the hierarchical approach again become the most feasible. Such examples could occur, e.g. if one relaxes the assumption of exponential lifetimes, cf. Section 5.

### *4.1 A hierarchical computation procedure*

This section presents a hierarchical procedure for calculating the dependability measure. Under this assumption, it is possible to group the states $i \in E$ into subclasses $E_0, E_1, \ldots, E_L$ such that any state $i \in E_k$ can only be entered from some $j \in \sum_{\ell=0}^{k-1} E_\ell$, and this decomposition then serves to provide a recursive scheme for computing the degradability measure. It should be noted that the decomposition $E=\sum_{\ell=0}^{L} E_\ell$ is a purely technical tool. As will be seen from the examples below, it has not necessarily any intrinsic interest and in particular it is not related in a natural way to the different decompositions given by the service levels.

We define $E_k$ as the set of states which can be reached in $k$ or less transitions. That is, $i \in E_k$ if there exists $i_0,\dots,i_{k-1} \in E$ such that

$$i_0 \neq i_1,\dots,i_{k-1}\neq i, \quad \pi_{i_0}\lambda_{i_0 i_1}\lambda_{i_1 i_2}\cdots\lambda_{i_{k-1}i} > 0,$$

and $k$ is the maximal number with this property. It is immediate that the $E_k$ are disjoint. Furthermore, $\lambda_{ij}=0$ whenever $i \in E_\ell$, $j \in E_k$ and $\ell \geq k$ (otherwise there is a chain of length $\ell+1$ leading to $j$ which contradicts $k$ being maximal) . The hierarchical assumption of no feedback amounts to

$$E=\sum_{\ell=0}^{L} E_\ell \tag{7}$$

for some $L$. This structure is found in the preceding example and would appear to cover a broad class of systems without repair or maintenance.

The following result shows that we can compute the $u_i$, $p_i$, $v_i$ recursively.

Theorem 1: *In the presence of the hierarchical structure (7),*

$$\begin{aligned} p_i &= \pi_i, \quad u_i=\frac{\pi_i}{\lambda_i}, \quad i\in E_0, && (8)\\ p_i &= \sum_{\ell=0}^{k-1}\sum_{j\in E_\ell:\lambda_{ji}>0} p_j\frac{\lambda_{ji}}{\lambda_j}, \quad i \in E_k, && (9)\\ u_i &= \frac{p_i}{\lambda_i}, \quad i\in E_k\cap O, && (10)\\ v_i &= \frac{1}{p_i}\sum_{j\in O} u_j, \quad i \in F. && (11)\end{aligned}$$

*Proof* Formula (8) is trivial. For (9), note that a $i \in E_k$ is necessarily entered from some state in $E_{k-1}$, say $j$. The probability of going to $j$ from $i$ is $\lambda_{ji}/\lambda_j$ and since $p_j$ represents the probability of ever entering $j$, (9) follows. The remaining formulas are exactly as before.

## *4.2 A computation method using matrix calculus*

Here we present a second algorithm that is based on a general matrix formalism. Let $\boldsymbol{\Lambda}$ be the $O\times O$ matrix with $ij$th entry $\lambda_{ij}$, $i,j\in O$. Note that $\boldsymbol{\Lambda}$ does not involve the $\lambda_{ji}$, $i \in F$; these are represented instead in terms of the (column) vectors $\boldsymbol{\ell}^{(i)}=(\lambda_{ji})_{j\in O}$, $i\in F$. We also let $\boldsymbol{e}^{(i)}$ denote the $i$th (column) unit vector and $\boldsymbol{q}^{(i)}$ the column vector $-\boldsymbol{\Lambda}^{-1}\boldsymbol{\ell}^{(i)}$. Note that the initial vector $\boldsymbol{\pi}$ is written as a row vector. Thus for example in (12) below, $\boldsymbol{\pi}\boldsymbol{\Lambda}^{-1}$ is a row vector and $\boldsymbol{\pi}\boldsymbol{\Lambda}^{-1}\boldsymbol{e}^{(i)}$ a real number (as should be).

The following result shows that once the key step of computing $\boldsymbol{\Lambda}^{-1}$ has been overcome, the performance measure can immediately be calculated:

Theorem 2:

$$\begin{aligned} u_i &= -\boldsymbol{\pi}\boldsymbol{\Lambda}^{-1}\boldsymbol{e}^{(i)}, \quad i\in O, && (12)\\ p_i &= u_i\lambda_i, \quad i \in O, && (13)\\ p_i &= -\boldsymbol{\pi}\boldsymbol{\Lambda}^{-1}\boldsymbol{\ell}^{(i)}, \quad i\in F, && (14)\\ v_i &= \frac{1}{p_i}\sum_{j\in O}u_j, \quad i\in F. && (15)\end{aligned}$$

*Proof* Formula (12) follows by noting that the $t$–step transition matrix of the Markov process is $e^{\boldsymbol{\Lambda}t}$, hence the vector of state probabilities at time $t$ is $\boldsymbol{\pi}e^{\boldsymbol{\Lambda}t}$ and the probability of being in $i$ is $\boldsymbol{\pi}e^{\boldsymbol{\Lambda}t}\boldsymbol{e}^{(i)}$, so that by standard formulas for integrating matrix–exponentials (Graham 1981)

$$u_i=\int_0^\infty \boldsymbol{\pi}e^{\boldsymbol{\Lambda}t}\boldsymbol{e}^{(i)}dt=-\boldsymbol{\pi}\boldsymbol{\Lambda}^{-1}\boldsymbol{e}^{(i)}.$$

Similarly, since the contribution to $p_i$ from the Markov process being in state $j$ in the time interval $[t,t+dt]$ is $\boldsymbol{\pi}e^{\boldsymbol{\Lambda}t}\boldsymbol{e}^{(j)}\cdot\lambda_{ji}dt$, we have

$$p_i=\sum_{j\in O}\int_0^\infty \boldsymbol{\pi}e^{\boldsymbol{\Lambda}t}\boldsymbol{e}^{(j)}\cdot\lambda_{ji}dt=\int_0^\infty \boldsymbol{\pi}e^{\boldsymbol{\Lambda}t}\boldsymbol{\ell}^{(i)}=.$$

.

$$=-\boldsymbol{\pi}\boldsymbol{\Lambda}^{-1}\boldsymbol{\ell}^{(i)}.$$

Formulas (13) and (15) follow from Theorem 1.

□

Example 2: We shall reinspect the formulas of Example 1 in view of Theorem 2. Recalling that $O=\{SL0,SL1\}$, we have

$$\Lambda=\begin{pmatrix} -\nu_A-\nu_B-\nu_C & \nu_A\\ 0 & -\nu_B-\nu_C\end{pmatrix},$$

$$\ell^{(SL2)}=\begin{pmatrix}\nu_B\\ \nu_B\end{pmatrix},\quad \ell^{(SL3)}=\begin{pmatrix}\nu_C\\ \nu_C\end{pmatrix}.$$

Thus

$$\Lambda^{-1}=\begin{pmatrix} -\frac{1}{\nu_A+\nu_B+\nu_C} & -\frac{\nu_A}{(\nu_A+\nu_B+\nu_C)(\nu_B+\nu_C)}\\ 0 & -\frac{1}{\nu_B+\nu_C}\end{pmatrix}.$$

Since $\pi = (1\ 0)$, $\pi\Lambda^{-1}$ is simply the first row of $\Lambda^{-1}$, which according to (12) is $(-u_{SL0} \quad -u_{SL1})$ — in accordance with the formulas of Example 1. Furthermore, $v_{SL2}$ for example can be calculated by noting that

$$p_{SL2} = \left( -\tfrac{1}{\nu_A+\nu_B+\nu_C} \quad -\tfrac{\nu_A}{(\nu_A+\nu_B+\nu_C)(\nu_B+\nu_C)} \right) \cdot$$

$$\cdot \begin{pmatrix} \nu_B \\ \nu_B \end{pmatrix} = \frac{\nu_B}{\nu_B+\nu_C},$$

□

## 5. MODELLING OF NON–EXPONENTIAL LIFETIMES

The analysis above is based on the assumption that the lifetimes of the components in the object system are exponentially distributed. This may often be quite unrealistic, especially when dealing with faults related to human interaction, such as software design faults. Using phase–type distributions instead of the exponential distribution allows us to dispense with this assumption at the expense of a higher complexity of the involved calculations. Still, phase–type assumptions give the possibility of remaining within the universe of Markovian modelling by introducing some additional states to the system. Furthermore, there is no essential loss of generality since any distribution can be approximated arbitrarily closely by a phase–type distribution.

### *5.1 Definition*

A random variable $Y$ is said to be *phase–type distributed* if it can be described as the time to absorption of a Markov process $\{J_t\}_{t\geq 0}$ with state space $G$ and intensity matrix $\boldsymbol{T}$. The Markov process must have fixed transition rates and $n$ states where one state is absorbing and all others are transient. If we let $\{n\}$ be the absorbing state, this means that $\lambda_{ni} = 0$ and $\lambda_i > 0$ for $1 \leq i \leq n-1$. Hence, we can never leave state $n$ but we must always leave the others sooner or later. By introducing the initial distribution $\boldsymbol{\pi}$ where $\pi_i = \mathbb{P}(J_0 = i)$, we can formally express $Y$ as $Y = \min\{t : J_t = n\}$.

This means that we can extend a Markov representation of a physical system by replacing any fixed state with a number of phase–type states and internal transition rates. Hence the sojourn times of the physical states become phase–type distributed rather than exponentially distributed.

Typically, a phase–type model of a lifetime is purely descriptive in the sense that there is no physical interpretations of the phases. Thus, the representation $(\boldsymbol{\pi}, \boldsymbol{T}, G)$ is obtained on the basis of empirical data of the lifetimes only. A recent software package is described in (Haggstrom *et al.* 1992) and the theoretical background is given in (Asmussen and Nerman 1991). An algorithm for estimating a phase–type distribution from censored samples is presented in (Olsson 1993). For a more complete exposition, see (Neuts 1981). Note that both the *hyperexponential distribution* and the *Erlang distribution* can be regarded as special cases of phase–type distributions.

### *5.2 Examples*

Example 3: Consider a software package consisting of one interactive application program module and one user interface program module. The purpose of the interface module is to protect the application program by checking that all data and commands entered by the user fulfill certain requirements stated by the application program, e.g. that no illegal data or command is entered. An illegal entry may cause the application program to crash. Assume that the software package has an exponential lifetime distribution with intensity $\lambda_1$, under the given trajectory in the command input space, (Littlewood 1988). However, due to a mistake in the program loading sequence, occurring say with probability $p$, the interface module may be missing so that all user input is simply transferred into the application module. Thus, the software package contains a *deficiency.* See (Jonsson 1993). If we adopt a conservative approach, we assume that the first illegal data/command input will then cause a program crash.

Suppose that the arrival intensity of illegal data/commands is $\lambda_2$, which typically is much larger than $\lambda_1$. We may take $O = \{1, 2\}$, 2 representing an unprotected application program, 1 a protected one, and $F$ consisting of a single state $f$. This leads to a simple hyperexponential distribution with a state diagram as given in Fig. 2, where the initial probabilities are $1-p$ and $p$ for states 1 and 2, respectively.

$$\begin{tikzcd} \boxed{1} \arrow[r, "\lambda_1"] & \boxed{f} & \boxed{2} \arrow[l, "\lambda_2"'] \end{tikzcd}$$

Figure 2. State diagram for deficient software

Using the notation of Section 4.2 we get

$$\pi = (1{-}p\ \ p), \quad \Lambda = \begin{pmatrix} -\lambda_1 & 0 \\ 0 & -\lambda_2 \end{pmatrix}, \quad \ell^{(f)} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}.$$

□

Example 4: Consider again the software package of the preceding example, but assume now that the lifetime distribution of the protected software module is much better approximated by an Erlang(2) distribution, say with intensity $\lambda_1$, than by an exponential distribution. Then we could split state 1 up into two states 0 and 1, representing the fictitious stages (which of the two exponential units are working) of the lifetime. We get the formulas

$$\pi = \ (1-p \ \ 0 \ \ p), \quad \Lambda = \begin{pmatrix} -\lambda_1 & \lambda_1 & 0 \\ 0 & -\lambda_1 & 0 \\ 0 & 0 & -\lambda_2 \end{pmatrix},$$

$$\ell^{(f)} = \begin{pmatrix} 0 \\ \lambda_1 \\ \lambda_2 \end{pmatrix}.$$

and the state diagram as given in Fig. 3, with initial probabilities $1-p$, 0 and $p$ for states 0, 1 and 2, respectively.

$$\begin{tikzcd} \boxed{0} \arrow[r, "\lambda_1"] & \boxed{1} \arrow[r, "\lambda_1"] & \boxed{f} & \boxed{2} \arrow[l, "\lambda_2"'] \end{tikzcd}$$

Figure 3. State diagram for phase–typed system

Note that the lifetime of such a system will typically have a failure rate with the commonly encountered 'bathtub' shape. □

In examples 3 and 4, it is the distribution of the lifetime of the system as a whole that is of phase–type. However, it is equally interesting to use non–exponential models for the lifetimes of individual components.

Example 5: In this example we shall apply the phase–type reasoning of example 3 and 4 to the process control system in example 1. See the corresponding state diagram in Fig. 4.

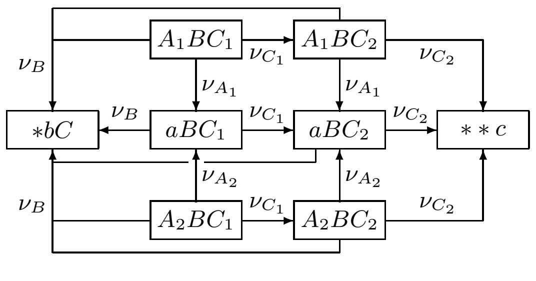

Figure 4. Process control system with two phase–type states

Suppose that the I/O computer $A$ with probability $p$ has deficient software according to example 3. Also suppose that the basic process support computer $C$ has been equipped with a cold redundancy so that the safety of the object system may be improved. A switchover to the redundant computer is automatically made in the case when the first one fails. The switchover mechanism is assumed to be ideal. This is modelled by an Erlang distribution according to the "left branch" of example 4. State $*bC$ is attained by pooling states $*bC_1$ and $*bC_2$. The service levels can readily be identified in the figure. For example, service level 0 is constituted by the two upper and the two lower boxes.

The entries in the dependability vector can be calculated using the following formulas:

$$\begin{aligned}
u_{A_1BC_1} &= p \cdot \frac{1}{\nu_{A_1}+\nu_B+\nu_{C_1}} \\
u_{A_2BC_1} &= (1-p) \cdot \frac{1}{\nu_{A_2}+\nu_B+\nu_{C_1}} \\
p_{aBC_1} &= p \cdot \frac{\nu_{A_1}}{\nu_{A_1}+\nu_B+\nu_{C_1}} + \\
&\quad +(1-p) \cdot \frac{\nu_{A_2}}{\nu_{A_2}+\nu_B+\nu_{C_1}} \\
u_{aBC_1} &= p_{aBC_1} \cdot \frac{1}{\nu_B+\nu_{C_1}} \\
p_{A_1BC_2} &= p \cdot \frac{\nu_{C_1}}{\nu_{A_1}+\nu_B+\nu_{C_1}} \\
p_{A_2BC_2} &= (1-p) \cdot \frac{\nu_{C_1}}{\nu_{A_2}+\nu_B+\nu_{C_1}} \\
u_{A_1BC_2} &= p_{A_1BC_2} \cdot \frac{1}{\nu_{A_1}+\nu_B+\nu_{C_2}} \\
u_{A_2BC_2} &= p_{A_2BC_2} \cdot \frac{1}{\nu_{A_2}+\nu_B+\nu_{C_2}} \\
p_{aBC_2} &= p_{A_1BC_2} \cdot \frac{\nu_{A_1}}{\nu_{A_1}+\nu_B+\nu_{C_2}} + \\
&\quad +p_{A_2BC_2} \cdot \frac{\nu_{A_2}}{\nu_{A_2}+\nu_B+\nu_{C_2}} \\
&\quad +p_{aBC_1} \cdot \frac{\nu_{C_1}}{\nu_B+\nu_{C_1}} \\
u_{aBC_2} &= p_{aBC_2} \cdot \frac{1}{\nu_B+\nu_{C_2}} \\
p_{*bC} &= \cdots = \frac{\nu_B}{\nu_B+\nu_{C_1}} \left(1+\frac{\nu_{C_1}}{\nu_B+\nu_{C_2}}\right) \\
p_{**c} &= \cdots = \frac{\nu_{C_1}}{\nu_B+\nu_{C_1}} \cdot \frac{\nu_{C_2}}{\nu_B+\nu_{C_2}} \\
v_{*bC} &= \cdots = \frac{1}{\nu_B} \\
v_{**c} &= \cdots = \frac{\nu_B+\nu_{C_1}+\nu_{C_2}}{\nu_{C_1}\nu_{C_2}}
\end{aligned}$$

Assume that the software in computer $A$ is deficient with a probability of $p = 0.05$ and that the degradation rate in this case is $\nu_{A2} = 1000/10000$. Also assume that the redundant computer $C$ has the same failure rate as the original one, i.e. $\nu_{C1} = \nu_{C2} = 1/10000$. With the remaining values unchanged, i.e. $\nu_{A1} = \nu_B = 9.5/10000$, we obtain the dependability vector

$$\mathbf{w} \ = \ (499 \ \ 544 \ \ 1053 \ \ 115000) \qquad (16)$$

hours. We note that the two operational entries are virtually unchanged. The entry for the "normal" failure is unchanged as expected. However, due to the insertion of a redundant computer $C$, the value for the catastrophic state is considerably higher than it previously was, i.e. the safety of the object system is improved. □

## 6. SUMMARY

We have taken a step toward a quantitative understanding of dependability. This has been done by merging attributes such as reliability, performability and safety into a more general quality and by defining a measure for it. A method to accommodate non-exponential failure rates for system components using phase–type distributions has also been suggested. Two different algorithms for the calculation of the measure have been given. The first one is appropriate for small systems and manual calculations. The second one, which makes use of matrix calculus, gives a general approach and is applicable to big systems and computer–based calculations.

## 7. ACKNOWLEDGEMENT

This work was supported in part by the PDCS2 project (Predictably Dependable Computing Systems) of the European ESPRIT/BRA program, under contract No. 90-02692P from the Swedish National Board for Industrial and Technical Development (NUTEK).

## REFERENCES

Asmussen, S. and O. Nerman (1991). 'Fitting phase-type distributions via the EM algorithm'. *Symposium i anvendt statistik, UNI-C, Copenhagen* pp. 335–346.

Beaudry, M. (1978). Performance-related reliability measures for computing systems.. In 'IEEE Transactions on Computers'. Vol. C-27.

Bobbio, A. and A. Cumani (1990). Ml estimations of the parameters of a PH distribution in canonical triangular form. Technical report. Instituto elettroteccnico nazionale galileo ferraris. Torino, Italy.

Graham, A. (1981). *Kronecker products and matrix calculus with applications.* Ellis Horwood, Chichester.

Haggstrom, O., S. Asmussen and O. Nerman (1992). 'EMPTH - a program for fitting phase-type distributions'. *Studies in statistical quality control and reliability.*

Heimann, D., N. Mittal and K.S. Trivedi (1991). Dependability modelling for computer systems. In 'Proceedings of the annual Reliability and Maintainality Symposium'. pp. 120–127.

Howard, R. (1971). *Dynamic probabilistic systems.* Wiley, ISBN 99-0002431-1. New York, USA.

Jonsson, E. (1993). A unified approach to dependability impaitments in computer systems. In 'IASTED international conference on reliability, quality control and risk assessment, Cambridge, MA'. IASTED-ACTA Press, ISBN 0-88986-181-1. pp. 173–178.

Jonsson, E. and T. Olovsson (1992). On the integration of security and dependability in computer systems. In 'Proceedings of the IASTED International Conference: Reliability, Control and Risk assessment'. ISBN 0-88986-171-4. IASTED. pp. 93–97.

Jonsson, E., M. Andersson and S. Asmussen (1994). An attempt to quantitative security modelling. Technical Report 178. Department of computer engineering. Goteborg, Sweden.

Laprie, J. (Ed.) (1992). *Dependability: Basic concepts and terminology.* ISBN 3-211-82296-8. Springer Verlag.

Littlewood, B. (1988). Software reliability modelling and identification. In S. Bittanti (Ed.). 'Lecture notes in computer science'. Vol. no. 341. Springer Verlag, Germany, ISBN 3-540-50695-0.

Meyer, J. (1980). On evaluating the performability of degradable computing systems. In 'IEEE Transactions on Computers'. Vol. C-29. pp. 720–731.

Meyer, J. F. (1992). Performability: a retrospective and some pointers to the future. In 'Performance evaluation'. Vol. 14. North-Holland Ltd, England, ISBN 0-7458-0613-9. pp. 139–156.

Neuts, M. (1981). *Matrix-Geometric Solutions in Stochastic Models.* John Hopkins University Press, Baltimore.

Olsson, M. (1993). Estimation of phase-type distributions from censored samples. Technical Report 36. Department of Mathematics, Chalmers University of Technology. Göteborg, Sweden.

Smith, R. and K.S. Trivedi (1987). A performability analysis of of two multi-processor systems. In 'Proc. 17th IEEE International Symposium on Fault-tolerant Computing, FTCS17'. Pittsburg, Pennsylvania. pp. 224–229.

Trivedi, K. (1982). *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* ISBN 0-13-711564-4. Prentice-Hall, New Jersey.