Operational Profiles for the Performability Evaluation of Multiuser Software*

John F. Meyer
EECS Department
The University of Michigan
Ann Arbor, MI USA

Effects of shared use on the performability of modular software can be evaluated by considering an operational environment which distinguishes the users. Specifically, we examine a total system consisting of a community of n users who share a software system with m modules, referred to simply as an *$n \times m$ system*. The computational demands of these users, as inherited by the modules, is represented by a continuous-time, finite-state Markov process X called the *operational profile* (see [1] for details concerning its justification). The profile's construction is based on the isolated profiles of individual users which, in the case of heterogeneous use, are pairwise-distinct Markov processes. Moreover, in the presence of other users, an individual profile can differ from its isolated version due to slowdowns caused by sharing. Further, due to design faults in the modules, failures may be experienced during execution. In this regard, shared use of a module can affect the rate at which a user experiences failures. Accordingly, we permit such rates to depend on both the state of the operational profile and the user in question. The extent to which a failure rate is altered by multiple use is called a *stress factor*, defined to be the ratio of that rate to the value experienced in isolation.

The combined profile-failure model thus accounts for effects of multiple use on both performance (module execution times) and dependability (failure times) in the presence of design faults. It is therefore naturally suited to evaluations of software performability. Further, since much of the model's complexity resides in the operational profile X, there are important questions as to how X might be reasonably constrained to permit feasible solutions of its steady-state distribution, particularly for large n and m. The latter are addressed here, where the main concepts and results can be summarized as follows.

As noted above, X is constructed by first considering the demands of individual users, as they would be experienced if use of the system was not shared. More precisely, for each user i $(1 \leq i \leq n)$ and each time $t \in T = [0, \infty]$, we let $W_{i,t}$ be the random variable

$$W_{i,t} = \begin{cases} j & \text{if user } i \text{ is executing module } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

where $1 \leq j \leq m$ and 0 means that user i is likewise passive at time t. (To avoid repeated references to the "0 otherwise" exception, passive use is identified with occupancy of a fictitious module named 0.) Then the *isolated profile of user i* is the stochastic process

$$W_i = \{W_{i,t} \mid t \in T\} \qquad (1)$$

where, with reasonable assumptions concerning single-user execution (again see [1]), W_i is a (time-homogeneous) Markov process. The (joint) *operational profile* is then the (Markov) process $X = \{X_t \mid t \in T\}$, where the variables X_t take values in the space

$$Q = \{0, 1, \ldots, m\}^n$$

of all n-tuples over the integers from 0 to m. A state $q = (q_1, q_2, \ldots q_n) \in Q$ is an *operational state*, with q_i being the module (perhaps the fictitious module 0) that's occupied by user i $(1 \leq i \leq n)$.

More precisely, the transition structure of X can be described as follows. Let $A_i = [a^i_{j,k}]$ be the transition-rate matrix (generator) of isolated profile W_i. A_i is thus an $(m+1)\times(m+1)$ matrix where, for $j \neq k$

$$\begin{aligned} a^i_{j,k} &= \text{the transition rate of } W_i \\ & \text{from module } j \text{ to module } k. \end{aligned} \qquad (2)$$

Then, for any pair of operational states $q = (q_1, q_2, \ldots, q_n)$ and $r = (r_1, r_2, \ldots, r_n)$ $(q, r \in Q)$, the transition-rate matrix $A = [a_{q,r}]$ of X has the following entries.

i) If q and r differ in more than one coordinate then

$$a_{q,r} = 0.$$

ii) If q and r differ in exactly one coordinate i $(1 \leq i \leq n)$, let $j = q_i$ and $k = r_i$. Then

$$a_{q,r} = a^i_{j,k}/s(q,i)\,, \qquad (3)$$

*This work was supported by a Visiting Fellowship Research Grant from the Engineering and Physical Sciences Research Council of the United Kingdom.

where $s(q,i)$ (a positive, real number qualified further below) is the *slowdown factor* (or simply *slowdown*) of user i in operational state q.

iii) If $q = r$ then

$$a_{q,q} = -\sum_{s \neq q} a_{q,s}.$$

Condition i) says that, at any given instant of time, at most one user can undergo a change in module occupancy. In view of the continuous-time nature of the isolated profiles, this is a reasonable assumption. Condition ii) refers to when user i causes an operational-state transition from q to r where, in state q, i occupies module j and, in state r, i occupies module j. In this case, the transition rate from q to r is obtained by dividing the $j \to k$ transition rate of W_i by the slowdown factor $s(q,i)$. Accordingly, if $s(q,i) > 1$ then the transition rate is indeed slower. On the other hand, if $0 < s(q,i) < 1$ then this factor reflects a "speedup" relative to the corresponding rate for user i's isolated profile. Since $s(q,i) = 1$ signifies an absence of slowdown (or speedup), a slowdown factor is *proper* if it has a value other than 1.

By projecting X onto individual coordinates of the state space, X determines n processes

$$X_i = \{X_{i,t} \mid t \in T\}, \qquad 1 \leq i \leq n$$

where $X_{i,t}$ is the ith coordinate of X_t. X_i is referred to as the *individual profile of user i* since, due to possible slowdowns caused by module sharing, X_i will generally differ from its corresponding isolated profile W_i. Indeed, assuming all operational states are reachable, it follows that $X_i = W_i$ if and only if, for any state q, $s(q,i) = 1$.

As discussed in [1], the transition rates defined by (3) permit more generality than warranted by the intended interpretation. Accordingly, the $s(q,i)$ can be naturally restricted as follows.

a) Since there is no contention in the fictitious module (module 0), for each user i and for all $q \in Q$,

$$\text{if } q_i = 0 \text{ then } s(q,i) = 1\,. \tag{4}$$

b) For each actual module $j > 0$, if $q_i = j$ then, presuming that slowdown of i is due only to those users who occupy module j in operational state q,

$$\begin{array}{l}\text{the value of } s(q,i) \text{ depends only}\\ \text{on the set } U(q,i) = \{\ell \mid q_\ell = q_i\}.\end{array} \tag{5}$$

Moreover, in the special case where i is the lone occupant of module $q_i = j$, it is reasonable to assume that there is no slowdown of i in state q, i.e.,

$$\text{if } U(q,i) = \{i\} \text{ then } s(q,i) = 1\,. \tag{6}$$

Concerning possible values of $s(q,i)$ when slowdown is proper ($s(q,i) \neq 1$), without further constraints, the only other requirement (per its definition; see (3)) is that $s(q,i)$ be some positive, real number. However, there are additional possible conditions which likewise arise quite naturally and serve, among other things, to reduce the number of different factor-values that need to be considered. For example, in many applications it may be the case that the value of a proper slowdown factor $s(q,i)$ depends only on the number of users that occupy module q_i, i.e., the size of $U(q,i)$, without regard to the identities of the sharing users. Presuming further that all users of q_i in state q are similarly affected, i.e., $q_k = q_i$ implies $s(q,k) = s(q,i)$, then the number of distinct proper slowdown-factor values associated with module q_i is at most $n-1$. Accordingly, the number of such values for an $n \times m$ profile is no greater than $(n-1)m$. When compared to an unconstrained profile (see (8) below), this number is relatively small.

A special case of the above, and perhaps the most natural situation regarding slowdown, is where $s(q,i)$ is <u>equal</u> to the number of users occupying module q_i, i.e, for each user i and each operational state q such that $q_i \neq 0$,

$$s(q,i) = |U(q,i)|\,. \tag{7}$$

Here, the set of proper slowdown-factor values is the same for each module, namely $\{2, \ldots, n\}$ (assuming $n \geq 2$); hence, the number of distinct proper values is reduced to $n-1$. In particular, this type of slowdown occurs in round-robin scheduling when the duration of the time slice becomes very small (referred to as "processor sharing" in queueing theory literature).

Turning now to the focus of this investigation, we examine the solution of the steady-state probability distribution of X with respect to consideration of the slowdown factors $s(q,i)$. As mentioned above, the slowdown factors of a general $n \times m$ operational profile X are relatively unconstrained, the only restrictions being those imposed by their intended interpretation (see conditions (4)-(6)). With this amount of freedom, the maximum number of different slowdown-factor values (other than the value 1 due to conditions (4) and (6)) can be as large as

$$nm(2^{n-1} - 1)\,. \tag{8}$$

This is a consequence of the fact that, for user i occupying an actual module j (fixing i, j and letting the operational state vary over all $q \in Q$ such that $q_i = j$), there are 2^{n-1} possibilities for the set of users, other than i, who occupy module j. In other words, in terms of the notation of condition (5), this is the number of user-sets of the form $U(q,i) - \{i\}$. Hence, by (5) and after eliminating the empty set, since it corresponds to condition (6) where $s(q,i) = 1$, the number of proper (i.e., $\neq 1$) slowdown-factor values can be as large as $2^{n-1} - 1$. Moreover, since this argument applies to any choice of an i, j pair (with $1 \leq i \leq n$ and $1 \leq j \leq m$), where different pairs may have disjoint value sets, the maximum number of values is that given by (8).

Aside from practical problems of dealing with this many parameter values, e.g., determining realistic estimates of each via experimentation with actual systems, there are also important concerns relating to model solution. Although what we ultimately seek to evaluate are designated measures of performability (and dependability, as in [1]), since these are formulated in terms of a combined profile-failure model, the latter must be admit to feasible solution methods. In particular, since the operational profile lies at the bottom of the total-system model, the solution of its module-occupancy distribution is a foremost consideration.

To this end we find that, with relatively mild restrictions, operational profiles can benefit from the known solution advantages of Markov processes which are "reversible" or, equivalently, satisfy a set of "detailed balance" equations (see [2, 3], for example). As defined, such processes are irreducible and recurrent. However, in the context of operational profiles, it is plausible to admit users who are *transient* in the sense that their isolated profiles have a transient state, e.g., once use becomes active, it is never again passive. Although this implies the existence of transient states for the (combined) profile X, if the recurrent states of X comprise a closed, irreducible subset of Q, we find that detailed balance likewise serves to characterize reversibility. In particular, this permits a system's performability, as experienced by some user i, to be examined at the extremes where use (in steady-state) by others is either always-active or always-passive.

Applying Kolmogorov's criteria (again see [2, 3]), necessary and sufficient conditions for a reversible $n \times 1$ profile are then established in terms of the slowdown factors. These conditions are relatively weak (as compared to processor sharing, for example), since they result in only a linear reduction of the maximum number of distinct value choices (8) (for large n, the fraction is approximately $2/n$). Moreover, by couching general $n \times m$ profiles in the setting of multiple-chain, closed queueing networks, it can be shown that such reversible individual-module profiles guarantee a product-form solution for the entire profile.

References

[1] J. F. Meyer, D. R. Wright, and B. Littlewood, "Dependability of modular software in a multiuser operational environment," in *Proc. 6th Int'l Symp. on Software Reliability Engineering*, pp. 170–179, Toulouse, France, October 1995.

[2] F. P. Kelly, *Reversibility and Stochastic Networks*, John Wiley, 1979.

[3] K. Kant, *Introduction to Computer System Performance Evaluation*, McGraw-Hill, 1992.