Proceedings of the International Conference on Dependable Systems and Networks (DSN 2002), Washington, DC, June 2002.

Performability Analysis of Guarded-Operation Duration: A Successive Model-Translation Approach[*]

Ann T. Tai[†] William H. Sanders[††] Leon Alkalai[‡] Savio N. Chau[‡] Kam S. Tso[†]

IA Tech, Inc.[†]
Los Angeles, CA 90024
{a.t.tai,k.tso}@ieee.org

University of Illinois[††]
Urbana, IL 61801
whs@crhc.uiuc.edu

Jet Propulsion Laboratory[‡]
Pasadena, CA 91109
{lalkalai,schau}@jpl.nasa.gov

Abstract

When making an engineering design decision, it is often necessary to consider its implications on both system performance and dependability. In this paper, we present a performability study that analyzes the guarded operation duration for onboard software upgrading. In particular, we define a "performability index" Y that quantifies the extent to which the guarded operation with a duration ϕ reduces the expected total performance degradation. In order to solve for Y, we progressively translate its formulation until it becomes an aggregate of constituent measures conducive to efficient reward model solutions. Based on the reward-mapping-enabled intermediate model, we specify reward structures in the composite base model which is built on three stochastic activity network reward models. We describe the model-translation approach and show its feasibility for design-oriented performability modeling.

1 Introduction

In order to protect an evolvable, distributed embedded system for long-life missions against the effects of design faults introduced by an onboard software upgrade, a methodology called *guarded software upgrading* (GSU) has been developed [1]. The GSU methodology is supported by a message-driven confidence-driven (MDCD) protocol that enables efficient use of checkpointing and acceptance test techniques for error containment and recovery. Specifically, the MDCD protocol ensures that the system functions properly after a software component is replaced by an updated version during a mission, while allowing the updated component to interact freely with other components in the system. The period during which the system is under the escort of the MDCD protocol is called "guarded operation."

Guarded operation thus permits an upgraded software component to start its service to the mission in a seamless fashion, and, if the escorting process determines that the upgraded component is not sufficiently reliable and thus imposes an unacceptable risk to the mission, ensures that the system will be safely downgraded back by replacing the upgraded software component with an earlier version. It is anticipated that sensible use of this escorting process will minimize the expected total performance degradation, which comprises 1) the performance penalty due to design-fault-caused failure, and 2) the performance reduction due to the overhead of the safeguard activities. Accordingly, an important design parameter is the duration of the guarded operation ϕ, as the total performance degradation is directly influenced by the length of the escorting process. In turn, this suggests that a performability analysis [2] is well-suited for the engineering decision-making.

Although we conducted separate dependability and performance studies for the MDCD protocol [3, 1], performability analysis with the above motivation presents us with new challenges. First, performability measures for engineering decision-making should be defined from a system designer's perspective, which naturally leads to a design-oriented formulation that may not be directly conducive to the final solution. Second, such a performability model usually covers a broad spectrum of system attributes which may have interdependencies among them. Those factors prevent us from evaluating the performability measure in a straightforward fashion (e.g., attempting to obtain the solution by directly mapping the measure to a single reward structure in a monolithic model).

To circumvent the difficulties, we propose an approach that solves the performability measure through successive model translation. In particular, we first define a "performability index" Y, that quantifies the extent to which the guarded operation with a duration ϕ reduces the expected total performance degradation, relative to the case in which guarded operation is completely absent. For clarity and simplicity of the design-oriented model, we allow Y to be formulated at a high level of abstraction. In order to solve for Y efficiently, we choose not to elaborate its formulation directly or map the design-oriented model to a

[*] The work reported in this paper was supported in part by Small Business Innovation Research (SBIR) Contract NAS3-99125 from the Jet Propulsion Laboratory, National Aeronautics and Space Administration.

monolithic, state-space based model. Instead, we apply analytic methods to translate the design-oriented model into an evaluation-oriented model that allows us to exploit efficient solution methods, successively closing the gap between the formulation of Y and its final solution.

More specifically, we begin with constructing a design-oriented model that formulates Y. We subsequently translate this design-oriented model, through analytic manipulation, into an evaluation-oriented form that is an aggregate of constituent measures conducive to reward model solutions. Based on this reward-mapping-enabled intermediate model, we take our final step to specify reward structures in the composite base model, which is built on three stochastic activity network (SAN) [4] reward models.

As with behavioral decomposition methods (see [5, 6], for example) and hierarchical composition techniques (see [7, 8], for example), the objective of this model-translation approach is to avoid dealing with a model that is too complex to allow derivation of a closed-form solution. The difference between those previously developed techniques and our approach is that we focus on translating a model progressively until it reaches a form that is a simple function of "constituent reward variables," each of which can be directly mapped to a reward structure for solution.

As described in Sections 3 and 4, when translation progresses, we are able to learn additional mathematical implications (to the performability measure) of the behavior of the system in question. By discovering them along the path of translation, we acquire ideas for efficient model construction and solution. This is an important advantage of the model-translation approach, because it supports performability studies of engineering problems in which mathematical properties or implications of the system behavior may not become apparent until we elaborate the formulation of the problem to a certain degree. Moreover, the process of transforming the problem of solving a complex performability measure into that of evaluating constituent reward variables naturally enables us to utilize efficient modeling techniques (such as behavioral decomposition and hierarchical composition) and modeling tools, which we might be unable to exploit without model translation.

The next section provides a review of the GSU methodology and guarded operation. Section 3 defines and formulates the performability measure. Section 4 describes the translation process, followed by Section 5, which shows how the reward structures are specified in SAN models. Section 6 presents an analysis of optimal guarded-operation duration. The paper is concluded by Section 7, which summarizes what we have accomplished.

2 Review of GSU Framework

The development of the GSU methodology is motivated by the challenge of guarding an embedded system against the adverse effects of design faults introduced by an onboard software upgrade [3, 1]. The performability study presented in this paper assumes that the underlying embedded system consists of three computing nodes. (This assumption is consistent with the current architecture of the Future Deliveries Testbed at JPL.) Since a software upgrade is normally conducted during a non-critical mission phase when the spacecraft and science functions do not require full computation power, only two processes corresponding to two different application software components are supposed to run concurrently and interact with each other. To exploit inherent system resource redundancies, we let the old version, in which we have high confidence due to its sufficiently long onboard execution time, escort the new-version software component through two stages of GSU, namely, *onboard validation* and *guarded operation*, as illustrated in Figure 1.

Further, we make use of the third processor, which would otherwise be idle during a non-critical mission phase, to accommodate the old version such that the three processes (i.e., the two corresponding to the new and old versions, and the process corresponding to the second application software component) can be executed concurrently. To aid in the description, we introduce the following notation:

$\mathrm{P}_1^{\mathrm{new}}$ The process corresponding to the new version of an application software component.

$\mathrm{P}_1^{\mathrm{old}}$ The process corresponding to the old version of the application software component.

P_2 The process corresponding to another application software component (which is not undergoing upgrade).

The first stage of GSU (onboard validation), which can be viewed as extended testing in an actual space environment, starts right after the new version is uploaded to the spacecraft. During this stage, the outgoing messages of the shadow process $\mathrm{P}_1^{\mathrm{new}}$ are suppressed but selectively logged, while $\mathrm{P}_1^{\mathrm{new}}$ receives the same incoming messages that the active process $\mathrm{P}_1^{\mathrm{old}}$ does. Thus, $\mathrm{P}_1^{\mathrm{new}}$ and $\mathrm{P}_1^{\mathrm{old}}$ can perform the same computation based on identical input data. By maintaining an onboard error log that can be downloaded to the ground for validation-results monitoring and Bayesian-statistics reliability analyses (as suggested by some prior work in the research literature, see [9], for example), we can make decisions regarding how long onboard validation should continue and whether $\mathrm{P}_1^{\mathrm{new}}$ can be allowed to enter mission operation. Moreover, onboard extended testing leads to a better estimation of the fault-manifestation rate of the upgraded software. If onboard validation concludes successfully, then $\mathrm{P}_1^{\mathrm{new}}$ and $\mathrm{P}_1^{\mathrm{old}}$ switch their roles to enter the guarded operation stage. The time to the next upgrade θ is determined upon the completion of onboard validation,

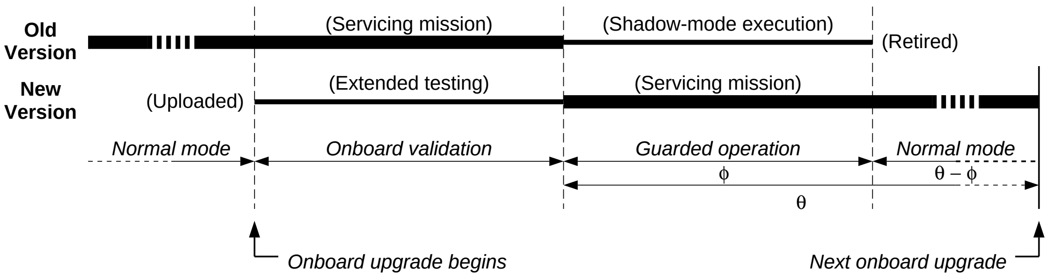

Figure 1: Onboard Guarded Software Upgrading

according to 1) the planned duty of the flight software in the forthcoming mission phases, and 2) the quality of the software learned through onboard validation.

During guarded operation, $\mathrm{P}_1^{\mathrm{new}}$ actually influences the external world and interacts with process P_2 under the escort of the MDCD error containment and recovery protocol, while the messages of $\mathrm{P}_1^{\mathrm{old}}$ that convey its computation results to P_2 or external systems (e.g., devices) are suppressed. We call the messages sent by processes to external systems and the messages between processes *external messages* and *internal messages*, respectively.

The key assumption used in the derivation of the MDCD algorithms is that an erroneous state of a process is likely to affect the correctness of its outgoing messages, while an erroneous message received by an application software component will result in process state contamination [3]. Accordingly, the necessary and sufficient condition for a process to establish a checkpoint is that the process receives a message that will make the process's otherwise non-contaminated state become potentially contaminated. In order to keep performance overhead low, the correctness validation mechanism, *acceptance test* (AT), is only used to validate external messages from the active processes that are potentially contaminated. By a "potentially contaminated process state," we mean 1) the process state of $\mathrm{P}_1^{\mathrm{new}}$ that is created from a low-confidence software component, or 2) a process state that reflects the receipt of a not-yet-validated message that is sent by a process when its process state is potentially contaminated.

Upon the detection of an erroneous external message, $\mathrm{P}_1^{\mathrm{old}}$ will take over $\mathrm{P}_1^{\mathrm{new}}$'s active role and prepare to resume normal computation with P_2. By locally checking its knowledge about whether its process state is contaminated, a process will decide to roll back or roll forward, respectively. After a rollback or roll-forward action, $\mathrm{P}_1^{\mathrm{old}}$ will "resend" the messages in its message log or further suppress messages it intends to send, based on the knowledge about the validity of $\mathrm{P}_1^{\mathrm{new}}$'s messages. After error recovery (which marks an unsuccessful but safe onboard upgrade), the system goes back to the normal mode (under which safeguard functions, namely, checkpointing and AT, are no longer performed) until the next scheduled upgrade. An undetected, erroneous external message[1] will result in system failure, meaning that the system will become unable to continue proper mission operation. On the other hand, as the MDCD algorithms allow error recovery to be trivial [1], we anticipate that the system will recover from an error successfully provided that the detection is successful.

If no error occurs during ϕ, then guarded operation concludes and the system goes back to normal mode at ϕ (see Figure 1). Note that while the time to the next scheduled onboard upgrade θ is chosen via a software engineering decision, the duration of guarded operation ϕ is a design parameter that influences system performance and dependability. The central purpose of this paper is to study how to evaluate a performability measure for determining an optimal ϕ. In the section that follows, we define and formulate the performability measure.

3 Performability Measure

3.1 Definition

We define a performability measure that will help us to choose the appropriate duration of guarded operation ϕ. More specifically, ϕ will be determined based on the value of the performability measure that quantifies the total performance degradation reduction resulting from guarded operation. As mentioned in Section 1, we consider two types of performance degradation, namely, 1) the performance degradation caused by the performance overhead of checkpoint establishment and AT-based validation, and 2) the performance degradation due to design-fault-caused failure.

Clearly, a greater value of ϕ implies 1) a decrease in the performance degradation due to the potential system failure caused by residual design faults in the upgraded software component, and 2) an increase in the performance degradation due to the overhead costs of checkpointing and AT. If we let W_ϕ denote the amount of "mission worth," which is quantified by the system time that is devoted to performing application tasks rather than the safeguard activities and

[1] For simplicity, in the remainder of the paper, we use the term "error" to refer to an erroneous external message.

accrued through θ when the duration of guarded operation (G-OP) is ϕ, then W_0 refers to the total mission worth accrued through θ for the boundary case in which the G-OP mode is completely absent (having a zero duration). On the other extreme, if the system is perfectly reliable, then it would not require G-OP and would thus be free of either type of performance degradation described above. We view this extreme case as the "ideal case" and let its total mission worth (accrued through θ) be denoted by W_{I}.

It is worthwhile noting that the difference between the expected values of W_{I} and W_ϕ can be regarded as the expected mission worth reduction, or the expected total performance degradation (from the ideal case) that the system experiences through θ when the G-OP duration is ϕ. Similarly, the difference between the expected values of W_{I} and W_0 represents the expected total performance degradation the system experiences through θ when the G-OP mode is absent throughout θ. It follows that if $E[W_{\mathrm{I}}] - E[W_\phi] < E[W_{\mathrm{I}}] - E[W_{\phi'}]$, then ϕ can be considered a better choice than ϕ'. Accordingly, we let the performability measure take the form of a *performability index* Y, that quantifies the extent to which a G-OP duration ϕ reduces the expected total performance degradation, relative to the case in which the G-OP mode is completely absent. More succinctly, Y is the ratio of the difference between $E[W_{\mathrm{I}}]$ and $E[W_0]$ to that between $E[W_{\mathrm{I}}]$ and $E[W_\phi]$:

$$Y = \frac{E[W_{\mathrm{I}}] - E[W_0]}{E[W_{\mathrm{I}}] - E[W_\phi]} \tag{1}$$

Based on the above discussion, we can anticipate performability benefit from a guarded operation that is characterized by a duration ϕ when $E[W_{\mathrm{I}}] - E[W_\phi]$ is less than $E[W_{\mathrm{I}}] - E[W_0]$. More precisely, $Y > 1$ implies that the application of guarded operation will yield performability benefit with respect to the reduction of total performance degradation. On the other hand, $Y \leq 1$ suggests that guarded operation will not be effective for total performance degradation reduction. We formulate $E[W_{\mathrm{I}}]$, $E[W_0]$, and $E[W_\phi]$ in the next subsection.

3.2 Formulation

As explained above, we choose to quantify "mission worth" in terms of the system time devoted to performing application tasks (rather than safeguard activities) that is accrued through mission period $[0, \theta]$. Further, the system behavior described in Section 2 suggests that an error that propagates to an external system will nullify the worth of that mission period. Since neither of the two cases, the ideal case and the case in which the G-OP mode is completely absent, involves safeguard activities, W_{I} and W_0 can be formulated in a straightforward fashion:

$$W_{\mathrm{I}} = 2\theta \tag{2}$$

$$W_0 = \begin{cases} 2\theta & \text{if no error occurs throughout } \theta \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Note that the coefficient 2 in the above equations is due to the fact that in the avionics system we consider, only two application processes actively service the mission during θ. (For the cases to which W_{I} and W_0 correspond, the two processes will always be $\mathrm{P}_1^{\mathrm{new}}$ and P_2.)

To help formulate an expression for W_ϕ, we group into several categories the possible behaviors (i.e., sample paths) that the system may take. In particular, since we do not make the assumption that $\mathrm{P}_1^{\mathrm{old}}$ and P_2 are perfectly reliable and AT has a full coverage, we must consider situations where the system fails during guarded operation, or fails after error recovery. This leads us to define three classes of sample paths: i) those in which no error occurs, and the system thus goes through the upgrade process successfully (called S1 below), ii) those in which an error occurs during $(0, \phi]$, and the system successfully recovers (called S2 below), and iii) those involving the occurrence of an error from which the system cannot recover (no mission worth is accumulated, so these paths are not considered in the expression of mission worth). More specifically, we define sets of sample paths S1 and S2 as follows:

S1) No error occurs by the end of ϕ, so the system enters the normal mode with $\mathrm{P}_1^{\mathrm{new}}$ and P_2 in mission operation after ϕ; the upgraded system subsequently goes through the period ($\theta - \phi$) successfully.

S2) An error occurs and is detected by $\mathrm{P}_1^{\mathrm{new}}$ or P_2 at τ, $0 < \tau \leq \phi$, so that error recovery brings the system into the normal mode with $\mathrm{P}_1^{\mathrm{old}}$ and P_2 in mission operation after τ; the recovered system subsequently goes through ($\theta - \tau$) successfully.

We let $\rho_{t,1}$ and $\rho_{t,2}$ denote the fractions of time during which $\mathrm{P}_1^{\mathrm{new}}$ and P_2 (respectively) make forward progress (rather than performing safeguard functions), given that the system is under the G-OP mode until t ($t \leq \phi$). Then, W_ϕ can be defined as follows:

$$W_\phi = \begin{cases} (\rho_{\phi,1} + \rho_{\phi,2})\phi + 2(\theta - \phi) & \text{if S1} \\ \gamma((\rho_{\tau,1} + \rho_{\tau,2})\tau + 2(\theta - \tau)) & \text{if S2} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where the coefficient γ ($0 < \gamma < 1$) is the *discount factor* that takes into account the additional mission worth reduction for an unsuccessful but safe onboard software upgrade, relative to the case in which the upgrade succeeds. We can define γ according to the implication of S2 for the system in question. For clarity of illustration, we will postpone our description of how we define γ until Section 6, in which we present the evaluation experiments and results.

In order to solve for Y, we first define a stochastic process, $\mathcal{X} = \{X_t \mid t \in [0, \theta]\}$, to represent the dynamics of the distributed embedded system that is undergoing an onboard upgrade. As mentioned in Section 2, when an error goes undetected, the system will lose its ability to continue mission operation, implying an absorbing state. If we let $\mathcal{A}_1$ denote the set of states of $\mathcal{X}$ in which no error has occurred in the system, then according to the definition of W_0,

$$E[W_0] = 2\theta\, P(X_\theta \in \mathcal{A}_1, \text{when G-OP duration is } 0) \quad (5)$$

Further, if we let $W_\phi^{\text{S1}} = (\rho_{\phi,1} + \rho_{\phi,2})\phi + 2(\theta - \phi)$ and $W_\phi^{\text{S2}} = \gamma((\rho_{\tau,1} + \rho_{\tau,2})\tau + 2(\theta - \tau))$ (see Eq. (4)), then we have,

$$E[W_\phi] = E[W_\phi^{\text{S1}}] + E[W_\phi^{\text{S2}}] \quad (6)$$

By definition, the stochastic process $\mathcal{X}$ will be in a state in $\mathcal{A}_1$ at θ if the system has a G-OP duration ϕ and experiences a sample path in S1. It follows that

$$\begin{aligned} E[W_\phi^{\text{S1}}] = &((\rho_{\phi,1} + \rho_{\phi,2})\phi + 2(\theta - \phi)) \times \\ &P(X_\theta \in \mathcal{A}_1, \text{when G-OP duration is } \phi) \end{aligned} \quad (7)$$

We notice that the application-purpose message-passing events that trigger checkpointing and AT (which dominate the performance overhead) are significantly more frequent than the fault-manifestation events. Moreover, the mean time between message-passing events is only seconds in length, whereas a reasonable value of ϕ will be in the range of hundreds or thousands of hours. Hence, we can assume that the system reaches a steady state with respect to the performance-overhead related events before an error occurs or the G-OP duration ends. Thus, $\rho_{t,1}$ and $\rho_{t,2}$ can be regarded as steady-state measures ρ_1 and ρ_2, respectively. Consequently, Eq. (7) becomes:

$$\begin{aligned} E[W_\phi^{\text{S1}}] = &((\rho_1 + \rho_2)\phi + 2(\theta - \phi)) \times \\ &P(X_\theta \in \mathcal{A}_1, \text{when G-OP duration is } \phi) \end{aligned} \quad (8)$$

The complexity of sample paths in S2, coupled with the fact that τ is a random variable that can assume a continuum of values, precludes the possibility of deriving a trivial expression for $E[W_\phi^{\text{S2}}]$. Accordingly, we let h be the probability density function (pdf) of τ, and f denote the pdf of the time to system failure that occurs after error recovery (when P_1^{old} and P_2 are in the mission operation). Then, $E[W_\phi^{\text{S2}}]$ can be formulated as follows:

$$\begin{aligned} E[W_\phi^{\text{S2}}] &= \gamma \int_0^\phi ((\rho_1 + \rho_2)\tau + 2(\theta - \tau)) \times \\ &\quad h(\tau) \left(1 - \int_\tau^\theta f(x)\, dx\right) d\tau \end{aligned} \quad (9)$$

4 Successive Model Translation

Next, we develop an approach that translates the formulation of Y progressively until it becomes a simple function of "constituent measures," each of which is ready to have a reward model solution. Figure 2 illustrates the process of successive model translation.

4.1 Translation toward Reward Model Solutions

As shown in Figure 2, the design-oriented formulation of Y results in some expressions (above the dashed line in the figure) that are at a high level of abstraction and cannot be solved directly. Therefore, we perform translation by applying analytic techniques to realize model decomposition and measure partition/conversion. In order to achieve solution efficiency, we let the stochastic process $\mathcal{X}$ be partitioned into two simpler processes, namely, $\mathcal{X}' = \{X'_t \mid t \in [0, \phi]\}$ and $\mathcal{X}'' = \{X''_t \mid t \in [0, \theta]\}$. The former represents the system behavior during the pre-designated G-OP interval[2]. The latter can represent the system behavior under the normal mode in two different situations: 1) after G-OP completes successfully, and 2) when the G-OP mode is completely absent during $[0, \theta]$. Furthermore, with a reasonably high message-sending rate, the likelihood that dormant error conditions will remain in a process state after error recovery is so low that the effect on system behavior is practically negligible [3]. This suggests that system behavior after an error recovery can be modeled in a way analogous to modeling the system behavior after a successful completion of G-OP (except that for the former case the two active components would be P_1^{old} and P_2). Hence, coupled with $\mathcal{X}'$, $\mathcal{X}''$ can also support the evaluation of dependability measures for the case that involves a successful error recovery.

The probabilities $P(X_\theta \in \mathcal{A}_1, \text{when G-OP duration is } 0)$ and $P(X_\theta \in \mathcal{A}_1, \text{when G-OP duration is } \phi)$ can then be solved in an efficient way. Specifically, the former can be converted into $P(X''_\theta \in \mathcal{A}''_1)$ while the latter can be translated as the product of $P(X'_\phi \in \mathcal{A}'_1)$ and $P(X''_{\theta-\phi} \in \mathcal{A}''_1)$, if we let $\mathcal{A}'_1$ and $\mathcal{A}''_1$ denote, respectively, the sets of states of $\mathcal{X}'$ and $\mathcal{X}''$ in which no error has occurred in the system. Consequently, we can solve each of those transient, instant-of-time measures by defining a reward structure in one of the decomposed models, as illustrated in Figure 2.

As explained in Section 3.2, we treat ρ_1 and ρ_2 as steady-state instant-of-time measures. This suggests that we can evaluate those two constituent measures in a reward model that represents the performance aspects of the stochastic process $\mathcal{X}'$ (and has no absorbing states). In other words, as illustrated in Figure 2, ρ_1 and ρ_2 are ready for reward model solutions, requiring no further translation.

Clearly, it is more challenging to translate the double integral in Eq. (9) into a form that is conducive to a reward

[2] The behavior of a recovered system within the interval $(\tau, \phi]$ can also be represented by $\mathcal{X}'$, if an error occurs and is detected at τ ($\tau \leq \phi$).

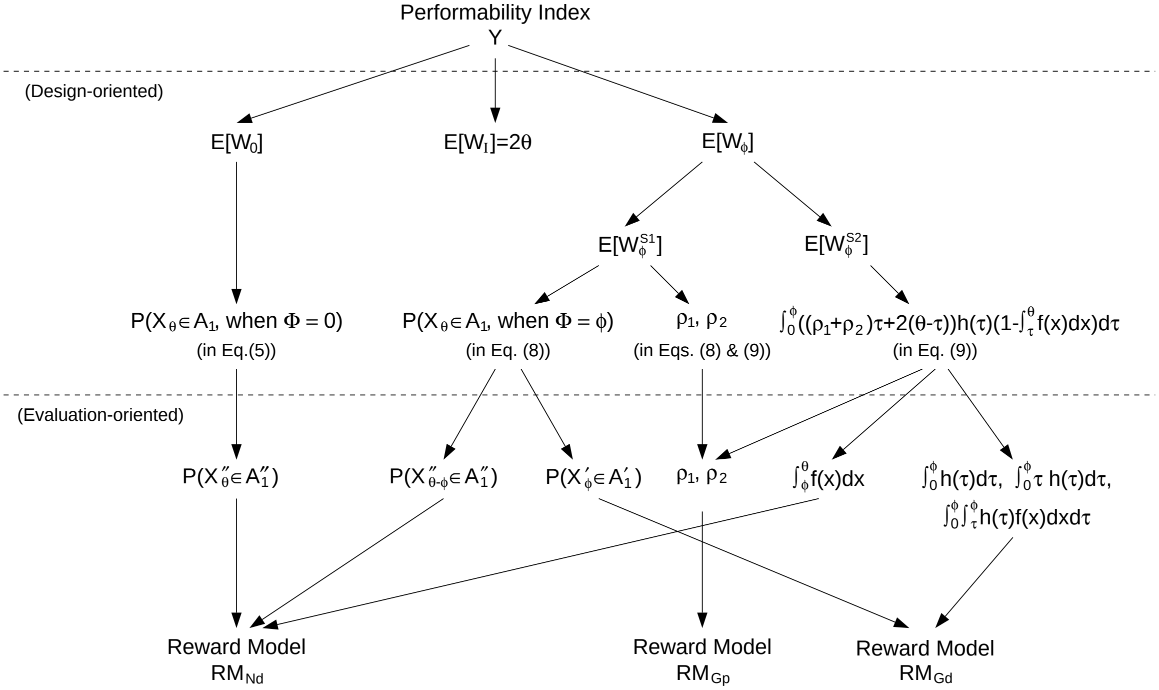

Figure 2: Successive Model Translation

model solution. Hence, we use judgment and make decisions regarding how to proceed along the path of translation. Step by step, as described in detail in Section 4.2, the double integral is converted into a form that is an aggregate of constituent measures, namely, ρ_1, ρ_2, $\int_0^\phi h(\tau)\,d\tau$, $\int_0^\phi \tau h(\tau)\,d\tau$, $\int_0^\phi \int_\tau^\phi h(\tau) f(x)\,dx\,d\tau$, and $\int_\phi^\theta f(x)\,dx$. For each of those measures, we can define a reward structure in one of the decomposed models that represents $\mathcal{X}'$ or $\mathcal{X}''$ and supports dependability or performance-overhead measures (as explained in the preceding paragraphs).

To this end, it becomes apparent that we will be able to solve for Y if we construct the following three reward models at the base-model level:

RM_{Gd} A reward model that represents the system behavior during the pre-designated G-OP interval and supports dependability measures.

RM_{Nd} A reward model that represents the system behavior under the normal mode and supports dependability measures.

RM_{Gp} A reward model that represents the system behavior under the G-OP mode and supports performance-overhead measures.

Details about the mapping between the resulting constituent measures and the reward structures in RM_{Gd}, RM_{Nd}, and RM_{Gp} are provided in Section 5.

4.2 Translation of $E[W_\phi^{\mathrm{S2}}]$

We now describe the translation process for the double integral in the expression for $E[W_\phi^{\mathrm{S2}}]$ (see Eq. (9)). We begin with rearranging its terms:

$$E[W_\phi^{\mathrm{S2}}] = \gamma \left(\int_0^\phi (2\theta - (2-(\rho_1+\rho_2))\tau) h(\tau)\, d\tau - \int_0^\phi (2\theta - (2-(\rho_1+\rho_2))\tau) h(\tau) \int_\tau^\theta f(x)\, dx\, d\tau \right) \quad (10)$$

If further, we rearrange the first term in the parentheses of Eq. (10), we have

$$\int_0^\phi (2\theta - (2-(\rho_1+\rho_2))\,\tau) h(\tau)\, d\tau = 2\theta \int_0^\phi h(\tau)\, d\tau - (2-(\rho_1+\rho_2)) \int_0^\phi \tau h(\tau)\, d\tau \quad (11)$$

Note that τ has a mixture distribution. This is because $h(\tau)$ equals zero for $\tau > \phi$ and thus $\lim_{\tau\to\infty} H(\tau) < 1$. Clearly, $\int_0^\phi h(\tau)\, d\tau$ is the probability that an error occurs and is detected by ϕ when the G-OP duration is ϕ. However, as mentioned earlier, the complexity of the system behavior makes it very difficult to derive h and compute the integrals without an excessive amount of approximation. Therefore, we choose to use reward model solution techniques and assume that rewards are associated with the states of $\mathcal{X}'$. More specifically, we let $\mathcal{A}'_3$ denote the set of states (of $\mathcal{X}'$) in which an error has occurred and been successfully detected, $\int_0^\phi h(\tau)\,d\tau$ can then be evaluated as the expected instant-of-time reward:

$$\int_0^\phi h(\tau)\,d\tau = P(X'_\phi \in \mathcal{A}'_3) \qquad (12)$$

In other words, with a state-space based model $\mathcal{X}'$, we can solve $\int_0^\phi h(\tau)\,d\tau$ by assigning a reward rate of 1 to all states in $\mathcal{A}'_3$ and a reward rate of zero to all other states, and computing the expected reward at ϕ.

Recall that the system behavior implies that 1) a state in which the system encounters an undetected error is absorbing, and 2) a successful error detection will result in error recovery that brings the system back to the normal mode under which checkpointing and AT (the error detection mechanism) will no longer be performed. In turn, this suggests that mean time to error detection $\int_0^\phi \tau h(\tau)\,d\tau$ is a meaningful measure, and it can have a reward model solution. Accordingly, if we let $\mathcal{A}'_2$ denote the set of states in which no error has been detected, and $\mathcal{A}'_4$ denote the set of (absorbing) states in which an error has occurred and caused a system failure due to unsuccessful error detection (thus $\mathcal{A}'_4$ is a proper subset of $\mathcal{A}'_2$), we have

$$\int_0^\phi \tau h(\tau)\,d\tau = \int_0^\phi (P(X'_t \in \mathcal{A}'_2) - P(X'_t \in \mathcal{A}'_4))\,dt \quad (13)$$

which implies that, to solve $\int_0^\phi \tau h(\tau)\,d\tau$, we can assign a reward rate of 1 to all states (of $\mathcal{X}'$) in $\mathcal{A}'_2$, a reward rate of -1 to all states in $\mathcal{A}'_4$, and a reward rate of zero to all other states, and then compute the expected reward accumulated through ϕ. Then, the integrals in Eq. (11) (and thus the first term of Eq. (10)) can be solved.

Next, we manipulate the second term in Eq. (10) in a similar fashion and begin with rearranging the terms:

$$\begin{aligned}
&\int_0^\phi (2\theta - (2-(\rho_1+\rho_2))\,\tau) h(\tau) \int_\tau^\theta f(x)\,dx\,d\tau \\
&= 2\theta \int_0^\phi \int_\tau^\theta h(\tau) f(x)\,dx\,d\tau - \\
&\quad (2-(\rho_1+\rho_2)) \int_0^\phi \int_\tau^\theta \tau h(\tau) f(x)\,dx\,d\tau \\
&\approx 2\theta \int_0^\phi \int_\tau^\theta h(\tau) f(x)\,dx\,d\tau \qquad (14)
\end{aligned}$$

We neglect the subtrahend because its value differs from those of θ and $E[W_\phi^{\mathrm{S2}}]$ by orders of magnitude. By carefully inspecting the area of the integration and changing the coordinates twice (due to space limitations, the details are omitted here but can be found in [10]), we break down the result of Eq. (14) into two terms, each of which can be interpreted in a straightforward fashion:

$$\begin{aligned}
2\theta \int_0^\phi \int_\tau^\theta h(\tau) f(x)\,dx\,d\tau &= 2\theta \int_0^\phi \int_\tau^\phi h(\tau) f(x)\,dx\,d\tau + \\
&\quad 2\theta \left(\int_0^\phi h(\tau)\,d\tau\right)\left(\int_\phi^\theta f(x)\,dx\right) \qquad (15)
\end{aligned}$$

The first summand in Eq. (15) can be interpreted as the probability that an error is detected when the system is under the G-OP mode and the recovered system fails by ϕ (under the normal mode) due to the occurrence of another error. A reward structure can then be defined accordingly in the reward model RM_{Gd} (which represents the dependability aspects of $\mathcal{X}'$). The second summand in Eq. (15) is indeed a product of two probabilities. While we have already interpreted $\int_0^\phi h(\tau)\,d\tau$ and proposed a reward model solution (see Eq. (12)), we recognize that $\int_\phi^\theta f(x)\,dx$ is the probability that the recovered system will fail due to the occurrence of another error at a time instant in $[\phi,\theta]$. As explained in Section 4.1, we can obtain a good approximation for $\int_\phi^\theta f(x)\,dx$ by defining a reward structure in RM_{Nd} (which represents the dependability aspects of $\mathcal{X}''$) and computing the expected instant-of-time reward at $(\theta-\phi)$.

To this end, we can evaluate each of the constituent measures of $E[W_\phi^{\mathrm{S2}}]$ by mapping it to a reward structure in RM_{Gd}, RM_{Nd}, or RM_{Gp}. In other words, if we plug the results of Eqs. (11) and (15) into Eq. (10), $E[W_\phi^{\mathrm{S2}}]$ becomes ready to be solved using reward model solution techniques.

5 SAN Reward Model Solutions for Constituent Measures

We use stochastic activity networks to realize the final step of model translation. By adopting and making necessary modifications to the SAN models we developed for our previous (separate) dependability and performance-cost studies [3, 1], we are able to use them as the reward models RM_{Gd}, RM_{Nd}, and RM_{Gp}. In the following, we briefly describe the SAN models and reward structures that support the evaluation of the constituent measures. Detailed descriptions are omitted here but can be found in [3, 1, 10].

5.1 SAN Reward Models

The SAN reward model RM_{Gd} is a modified version of the model we built for studying the dependability gain from the use of the MDCD protocol [3]. Modifications are made so that the model explicitly represents whether an error has been detected in the system; thus, each of the constituent measures $\int_0^\phi h(\tau)\,d\tau$, $\int_0^\phi \tau h(\tau)\,d\tau$, and $\int_0^\phi \int_\tau^\phi h(\tau) f(x)\,dx\,d\tau$ can be easily mapped to a reward structure. In model construction, we avoid modeling details about checkpoint establishment, deletion, and rollback

Table 1: Constituent Measures and SAN Reward Structures in RM_{Gd}

Measure	Reward Type	Predicate-Rate Pair	
$\int_0^{\phi} h(\tau)\, d\tau$	Expected instant-of-time reward at ϕ	`MARK(detected)==1 && MARK(failure)==0`	1
$\int_0^{\phi} \tau h(\tau)\, d\tau$	Expected accumulated interval-of-time reward for $[0, \phi]$	`MARK(detected)==0`	1
		`MARK(detected)==0 && MARK(failure)==1`	-1
$\int_0^{\phi} \int_{\tau}^{\phi} h(\tau) f(x)\, dx\, d\tau$	Expected instant-of-time reward at ϕ	`MARK(detected)==1 && MARK(failure)==1`	1
$P(X'_{\phi} \in \mathcal{A}'_1)$	Expected instant-of-time reward at ϕ	`MARK(detected)==0 && MARK(failure)==0`	1

Table 2: Constituent Measures and SAN Reward Structures in RM_{Gp}

Measure	Reward Type	Predicate-Rate Pair	
$1 - \rho_1$	Expected instant-of-time reward at steady state	`MARK(P1nExt)==1`	1
$1 - \rho_2$	Expected instant-of-time reward at steady state	`(MARK(P1nInt)==1 && MARK(P2DB) == 0) ||` `(MARK(P2Ext)==1 && MARK(P2DB) == 1)`	1

error recovery. Rather, by exploiting the relations among the markings of the places that represent whether a process is actually error-contaminated and the process's knowledge about its state contamination, we are able to characterize the system's failure behavior precisely with respect to whether messages sent by potentially contaminated processes will cause system failure.

In contrast, in the SAN reward model RM_{Gp}, we omit those failure-behavior-related aspects, such as error occurrence and unsuccessful error detection [1]. Instead, we focus on representing those conditions that would trigger a process to take actions that will cause the system to incur overhead costs (e.g., the action to establish a checkpoint, or to perform an AT). The SAN reward model RM_{Nd} is rather trivial; the illustration of this model can be found in [10].

5.2 SAN Reward Structures

As a result of model translation, each of the constituent measures is in a form that can be easily mapped to a reward structure in one of the SAN reward models. In addition, the *UltraSAN* tool provides us with a convenient way to define a reward structure by specifying a "predicate-rate" pair [11]. Below we describe how the reward structures are specified in each of the SAN reward models.

As indicated in Figure 2 and explained in Section 4.1, three constituent measures are supposed to be solved in the reward model RM_{Nd}, namely, $P(X''_{\theta} \in \mathcal{A}''_1)$, $P(X''_{\theta-\phi} \in \mathcal{A}''_1)$, and $\int_{\phi}^{\theta} f(x)\, dx$. To solve $P(X''_{\theta} \in \mathcal{A}''_1)$ and $P(X''_{\theta-\phi} \in \mathcal{A}''_1)$, we assign the fault-manifestation rate of P_1^{new} to the activity that represents the fault-manifestation behavior of the first software component, and compute the expected reward values at θ and $(\theta - \phi)$, respectively. As for $\int_{\phi}^{\theta} f(x)\, dx$, since it can be treated as the probability that the recovered system (consisting of P_1^{old} and P_2) fails during the interval $[0, \theta - \phi]$, we assign the fault-manifestation rate of P_1^{old} to the activity that represents the fault-manifestation behavior of the first software component, and compute the complement of the expected reward value at $(\theta - \phi)$. Due to the similarity among these constituent measures, the expected instant-of-time reward values explained above can be evaluated using the same predicate-rate pair:

- *Predicate:* `MARK(failure) == 0`
- *Rate:* `1`

Also as indicated in Figure 2, the constituent measures $\int_0^{\phi} h(\tau)\, d\tau$, $\int_0^{\phi} \tau h(\tau)\, d\tau$, $\int_0^{\phi} \int_{\tau}^{\phi} h(\tau) f(x)\, dx\, d\tau$, and $P(X'_{\phi} \in \mathcal{A}'_1)$ are supposed to be evaluated in the reward model RM_{Gd}. Table 1 summarizes how reward structures are specified in predicate-rate pairs and how the expected reward values are computed for solving those constituent measures. An explanation of those reward model solutions has been given in Section 4.

Finally, as indicated in Figure 2, two constituent measures are supposed to be solved in the reward model RM_{Gp}, namely, ρ_1 and ρ_2. For simplicity and clarity of the specification of the predicate-rate pairs, we instead solve for $(1 - \rho_1)$ and $(1 - \rho_2)$, which are the performance overhead measures for P_1^{new} and P_2, respectively. Table 2 enumerates the reward type and predicate-rate pair for each of the two measures.

6 Evaluation Results

Applying the SAN reward models and *UltraSAN*, we evaluate the performability index Y. Before we proceed to discuss the numerical results, we define the following notation:

μ_{new} Fault-manifestation rate of the process corresponding to the newly upgraded software version.

μ_{old} Fault-manifestation rate of a process corresponding to an old software version.

c Coverage of an acceptance test.

λ Message-sending rate of a process.

p_{ext} Probability that the message a process intends to send is an external message.

α Acceptance-test completion rate.

β Checkpoint-establishment completion rate.

We begin with conducting a study on the optimality of G-OP duration ϕ, considering the impact of the fault-manifestation rate of the upgraded software component. Specifically, we assume $\theta = 10000$ and use the parameter values shown in Table 3, in which all the parameters involving time presume that time is quantified in hours. Accordingly, $\lambda = 1200$ means that the time between message sending events (for an individual process) is 3 seconds; similarly, $\alpha = 6000$ and $\beta = 6000$ imply that the mean time to the completion of an AT-based validation and the mean time to the completion of a checkpoint establishment are both 600 milliseconds. Further, we let γ (see Eq. (4)) be a decreasing function of $\overline{\tau}$, the mean time to error detection. More succinctly, $\gamma = 1 - \frac{\overline{\tau}}{\theta}$. This function is defined based on the following consideration. Safeguard activities would no longer be performed after τ when error detection brings the system back to the normal mode with P_1^{old} and P_2 in mission operation; since that implies an unsuccessful (but safe) onboard upgrade, the performance cost paid for the safeguard activities up to τ would yield an additional reduction of mission worth, relative to the case of a successful onboard upgrade.

Table 3: Parameter Value Assignment

λ	μ_{new}	μ_{old}	c	P_{ext}	α	β
1200	10^{-4}	10^{-8}	0.95	0.1	6000	6000

The numerical results from this study are displayed by the curve with solid dots in Figure 3. The values of the performability index indicate that the optimal duration of the G-OP mode for this particular setting is 7000 hours, which yields the best worth of mission period θ, due to the greatest possible reduction of expected total performance degradation. This implies that for this particular setting, a ϕ smaller than 7000 would lead to a greater expected performance degradation due to the increased risk of potential design-fault-caused failure. On the other hand, if we let ϕ be larger, then the increased performance degradation due to the increased performance costs would more than negate the benefit from the extended guarded operation.

By decrementing the fault-manifestation rate of P_1^{new} (μ_{new}) to 0.5×10^{-4} (while letting other parameter values remain the same), we obtain another set of values of the performability index, as illustrated by the companion curve marked by hollow dots in Figure 3. The two curves together reveal that the optimality of ϕ is very sensitive to the reliability of the upgraded software component. In particular, we observe that when μ_{new} is decremented from 10^{-4} to 0.5×10^{-4}, the optimal ϕ is dropped from 7000 to 5000 (hours), even though the performance costs of safeguard activities remain low (thus ρ_1 and ρ_2 remain high, and equal 0.98 and 0.95, respectively). While it is quite obvious that a smaller μ_{new} will favor a shorter duration of the G-OP mode, this study confirms the relation between the two system attributes and helps us to recognize the sensitivity of this relation.

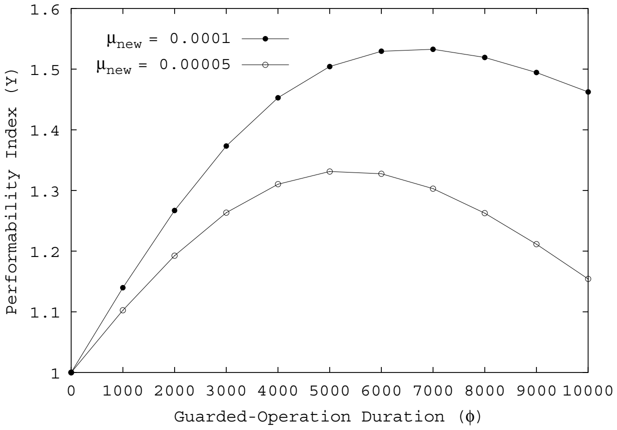

Figure 3: Effect of μ_{new} on Optimal ϕ ($\theta = 10000$)

As described in Section 2, θ is indeed chosen based on a software engineering decision (at the time onboard validation completes); the decision depends upon at least two factors: 1) the planned duty of the flight software in the forthcoming mission phases, and 2) the quality of the flight software learned through onboard validation. Hence, in the next study, we analyze the relationships between the values of θ and the optimal ϕ. Specifically, we repeat the study that yields the results shown in Figure 3, but letting the value of θ be reduced to 5000 hours. The resulting curves are displayed in Figure 4.

It is interesting to observe that, while other parameter values remain the same (meaning that the performance and dependability attributes of the system itself do not differ from those assumed in the previous study), the reduction of θ significantly changes the values for the optimal ϕ. Specifically, the optimal values of ϕ for the cases in which μ_{new}

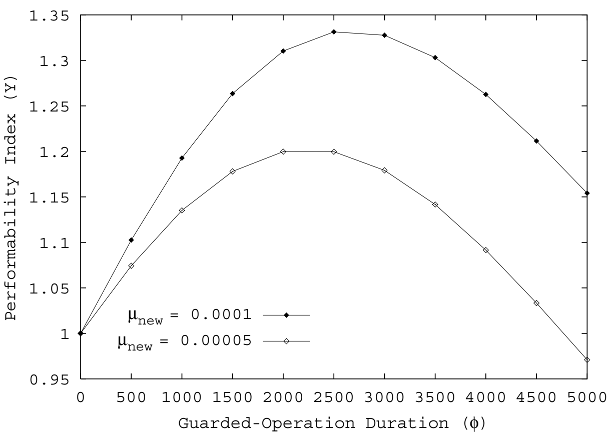

Figure 4: Effect of μ_{new} on Optimal ϕ ($\theta = 5000$)

equals 10^{-4} and 0.5×10^{-4} go down to 2500 and 2000, respectively. This can be understood by considering that reliability is generally a decreasing function of time, when maintenance is not available for a system. More precisely, when the anticipated time to the next onboard upgrade becomes shorter, the likelihood that the system will fail before the forthcoming upgrade decreases, permitting guarded operation to end at an earlier point to minimize the expected total performance degradation. By inspecting the results of the constituent measures that are available to us, namely, $P(X''_\theta \in \mathcal{A}''_1)$ and $\int_0^\phi h(\tau)\, d\tau$, we are able to validate this explanation.

We have also studied the impact on the optimal G-OP duration of the performance cost of safeguard functions (by varying the values of α and β). The results again exemplify the tradeoffs between the two types of expected performance degradation. More specifically, the study demonstrates that an increase in performance overhead tends to further negate the dependability benefits from safeguard functions, and thus will suggest an earlier cutoff line for guarded operation.

7 Concluding Remarks

We have conducted a model-based performability study that analyzes the guarded-operation duration for onboard software upgrading. By translating a design-oriented model into an evaluation-oriented model, we are able to reach a reward model solution for the performability index Y that supports the decision on the duration of guarded operation.

The successive model-translation approach enables us to expose hidden opportunities to apply efficient model construction/solution strategies and modeling tools. More specifically, this approach has a unique advantage: it enables us to conduct performability analyses for solving engineering problems in which boundaries and/or relationships among the system attributes involved in a performability measure are not obvious (from a mathematical point of view) and thus traditional reward model solution techniques, behavioral decomposition and hierarchical composition methods are not directly applicable.

Moreover, since its goal is to transform the problem of solving a performability measure into that of evaluating constituent reward variables, the model-translation approach permits us to access the results of the constituent measures to gain more insights from a model-based performability evaluation.

References

[1] A. T. Tai, K. S. Tso, L. Alkalai, S. N. Chau, and W. H. Sanders, “Low-cost error containment and recovery for onboard guarded software upgrading and beyond,” *IEEE Trans. Computers*, vol. 51, pp. 121–137, Feb. 2002.

[2] J. F. Meyer, “On evaluating the performability of degradable computing systems,” *IEEE Trans. Computers*, vol. C-29, pp. 720–731, Aug. 1980.

[3] A. T. Tai, K. S. Tso, L. Alkalai, S. N. Chau, and W. H. Sanders, “On the effectiveness of a message-driven confidence-driven protocol for guarded software upgrading,” *Performance Evaluation*, vol. 44, pp. 211–236, Apr. 2001.

[4] J. F. Meyer, A. Movaghar, and W. H. Sanders, “Stochastic activity networks: Structure, behavior, and application,” in *Proc. Int’l Workshop on Timed Petri Nets*, (Torino, Italy), pp. 106–115, July 1985.

[5] R. Geist, “Extended behavioral decomposition for estimating ultrahigh reliability,” *IEEE Trans. Reliability*, vol. R-40, pp. 22–28, Apr. 1991.

[6] G. Ciardo and K. S. Trivedi, “A decomposition approach for stochastic reward net models,” *Performance Evaluation*, vol. 18, no. 1, pp. 37–59, 1993.

[7] M. Veeraraghavan and K. S. Trivedi, “Hierarchical modeling for reliability and performance measures,” in *Concurrent Computations* (S. K. Tewsburg, B. W. Dickinson, and S. C. Schwartz, eds.), pp. 449–474, Plenum Publishing Corporation, 1988.

[8] M. Malhotra and K. S. Trivedi, “A methodology for formal expression of hierarchy in model solution,” in *Proc. the 5th International Workshop on Petri Nets and Performance Models*, (Toulouse, France), pp. 258–267, Oct. 1993.

[9] B. Littlewood and D. Wright, “Stopping rules for the operational testing of safety-critical software,” in *Digest of the 25th Annual International Symposium on Fault-Tolerant Computing*, (Pasadena, CA), pp. 444–453, June 1995.

[10] A. T. Tai and K. S. Tso, “On-board maintenance for affordable, evolvable and dependable spaceborne systems,” SBIR Phase-II Final Technical Report for Contract NAS3-99125, IA Tech, Inc., Los Angeles, CA, Mar. 2002.

[11] W. H. Sanders, W. D. Obal II, M. A. Qureshi, and F. K. Widjanarko, “The *UltraSAN* modeling environment,” *Performance Evaluation*, vol. 24, no. 1, pp. 89–115, 1995.