A Recurrence-Relation-Based Reward Model for Performability Evaluation of Embedded Systems

Ann T. Tai Kam S. Tso
IA Tech, Inc.
Los Angeles, CA 90094
{a.t.tai,k.tso}@ieee.org

William H. Sanders
University of Illinois
Urbana, IL 61801
whs@uiuc.edu

Abstract

Embedded systems for closed-loop applications often behave as discrete-time semi-Markov processes (DTSMPs). Performability measures most meaningful to iterative embedded systems, such as accumulated reward, are thus difficult to solve analytically in general. In this paper, we propose a recurrence-relation-based (RRB) reward model to evaluate such measures. A critical element in RRB reward models is the notion of state-entry probability. This notion enables us to utilize the embedded Markov chain in a DTSMP in a novel way. More specifically, we formulate state-entry probabilities, state-occupancy probabilities, and expressions concerning accumulated reward solely in terms of state-entry probability and its companion term, namely the expected accumulated reward at the point of state entry. As a result, recurrence relations abstract away all the intermediate points that lack the memoryless property, enabling a solvable model to be directly built upon the embedded Markov chain. To show the usefulness of RRB reward models, we evaluate an embedded system for which we leverage the proposed notion and methods to solve a variety of probabilistic measures analytically.

1. Introduction

Discrete-time performability analysis in terms of accumulated reward is well-suited to modeling of embedded systems, because they typically execute in open or closed loops or cycles[1], each of which accommodates at most one control command update. Further, the notions of "return" and "duty cycle" [1] (the latter refers to the proportion of time a system actively makes progress toward its task goal) promote the measures concerning expected accumulated reward.

Nonetheless, performability modeling for those systems can be difficult. In particular, while embedded systems have relatively simple architectures and functionalities, the behavior of an embedded system is often non-Markovian in nature. For example, an application may require its host to be engaged in a specific operation through a pre-designated time interval or to take a particular action with a specified frequency, which implies a deterministic sojourn time or a nonstationary transition probability, respectively.

[1]In the remainder of the text, the words "loop," "cycle," "frame," and "iteration" are used interchangeably.

While those non-Markovian properties can be circumvented using the notion of embedded Markov chain (as such a process behaves just like an ordinary Markov process at the points of state transition), performability measures based on accumulated reward can still be difficult to solve analytically. In addition, embedded applications may involve time- or path-dependent behavior, which may prevent a reward model from being analytically manageable.

Prior work in solving reliability, performance, and performability measures for discrete-time semi-Markov processes (DTSMPS) relied largely on simulation tools (see [2], for example), empirical, measurement-based estimation (see [3], for example), nonparametric approach (see [4], for example), and numerical approximation (see [5], for example). Efforts for analytic solutions of DTSMP were significantly less and were generally limited to models with restrictive assumptions (see [6], for example). Although simulation-based modeling tools, empirical approaches, and numerical-approximation methods are flexible and powerful, they may lose their advantages when a user desires to obtain insights from explicit expressions, such as a reachability graph and a symbolic solution which are unlikely to be supplied by a simulation or numerical-approximation tool.

In our earlier efforts, we leveraged recurrence relations for reliability assessment of a fault-tolerant bus architecture for an avionics system [7]. In addition, we used recurrence-relations to evaluate a distributed embedded system [8]. Recently, we have revisited the recurrence-relation-based approach and generalized it so that the framework can be applied to an important class of embedded systems, namely closed-loop or iterative applications. The central purpose is to enable a system or a design to be evaluated in a model-based way to complement the methods for simulation-based or testbed-enabled assessments.

For this purpose, we introduce a *recurrence-relation-based (RRB) reward model* in this paper. A critical element in RRB reward models is the notion of *state-entry probability*. Informally speaking, it is the probability that the system in question will enter into a specific state at a particular cycle. This notion enables us to utilize the embedded Markov chain in a DTSMP in a novel way. More specifically, in a recurrence relation derivation, we formulate state-entry probabilities themselves and state-occupancy probabilities solely in terms of state-entry probability. Likewise, we let the expressions concerning expected accumulated reward be formulated solely in terms of state-entry probability and its companion term, namely expected reward at the point of state entry. As a result, recurrence relations naturally abstract away all the intermediate points (that constitute a deterministic sojourn time or a fixed interval between two consecutive occurrences of a periodic event) that lack the memoryless property, enabling a solvable model to be directly built upon the embedded Markov chain.

The remainder of the paper is organized as follows. Section 2 provides background information on closed-loop embedded systems and discusses the feasibility of using an RRB approach. Section 3 introduces the notion of state-entry probability and describes how it enables the construction and solution of an RRB model. Section 4 presents a sample application to exemplify the applicability of RRB reward models to embedded systems and the types of measure that can be solved analytically using such a reward model. We conclude the paper in Section 5.

2. System Model & RRB Approach Feasibility

A majority of embedded systems make control, command, and other types of dynamic decisions. Those decisions are usually made by a software subsystem and based on the feedback from the underlying hardware subsystem. Typically, a sample from the sensors represents position, voltage, temperature, or other parameters. Each sample provides the software subsystem with updated information which is called *feedback*. Systems that make use of feedback are called *closed-loop embedded systems*. Familiar examples of closed-loop embedded systems include those driving thermostats and automobile cruise control. If feedback indicates that the room temperature is below the desired setpoint, the thermostat will turn the heater on until that temperature is reached. Similarly, if a car is going too quickly, the cruise-control system can temporarily reduce the amount of fuel fed to the engine. In both cases, the feedback enables compensations for disturbances to the system (such as changes in the outdoor temperature and in the roughness or steepness of the road).

Sophisticated embedded applications include robotics systems that service outer-planet exploration and make autonomous decisions to ensure the accomplishment of a mission. For such systems, the control basis provides a discrete state representation that reflects the status of a mission task. Closed-loop controllers ensure asymptotically stable system behavior that is robust to local perturbations, while perturbations are learned from the feedback. Moreover, action-selection policies are often defined on a Markov decision process to find and track a visual feature. In general, the objective of closed-loop system operation is to minimize control inaccuracy and the gap between the system's actual behavior and its functional specification.

Closed-loop embedded systems typically operate at a fixed frequency. The frequency of command update that drives the hardware subsystem usually follows the sampling rate (i.e., bounded above by the sampling rate). After reading the most recent sample from the sensors, the software subsystem reacts to the learned system-state change by calculating the required adjustments and issuing a new command. The hardware subsystem then responds to the new command, followed by another sample-taking. Then the cycle repeats. Eventually, when the system reaches a desired state or its mission period ends, the software subsystem will cease calculating and issuing commands.

Moreover, in advanced closed-loop embedded systems, reinforced learning is achieved by letting a software agent that makes decisions for adaptation see a "return" for each cycle. The return has a numerical value that encodes the extent of the success (or failure) of an action and is learned by the agent in retrospect. Accordingly, the agent will become increasingly capable of selecting actions that maximize the accumulated return over time.

Together with the notions of duty cycle and non-Markovian behavior mentioned in Section 1, the above system properties can be summarized as follows:

P1) The frequency of decision/action is bounded above by the sampling rate, which implies that each cycle accommodates at most one state transition.

P2) Each of the successive iterations takes the most recent feedback from sensors as the parameters for calculating the required adjustments for the subsequent operation cycle.

P3) The notion of duty cycle can be adopted to quantify the extent of a mission's success.

Those system properties collectively suggest a mathematical model for iterative embedded systems. Specifically, coupled with embedded systems' non-Markovian behavior, P1 suggests that a closed-loop system can be represented as a discrete-time semi-Markov process; P2 implies that an efficient way to model a system's path-/time-dependent behavior or error propagation is to leverage recurrence relations that allow systematic backward tracking; and P3 means that a modeling framework for iterative embedded systems should explicitly support the solution of accumulative reward and time-averaged accumulative reward for quantifying an embedded system's gracefully degradable performance. Based on those observations, we elaborate our RRB approach in the following section.

3. RRB Reward Model

3.1. Formulation of State-Entry Probability

Letting $\mathcal{M}$ and Π denote the system in question and its state space, respectively, then state-entry probability $\widehat{P}_k[i]$ can be defined as follows:

Definition 1 $\widehat{P}_k[i]$ *is the probability that* $\mathcal{M}$ *will enter* S_k *in the* i*th cycle from* $\{S_j \mid j \in \Pi - \{k\}\}$.

Without losing generality, the transition probability p_{jk} for each of the states j that is not connected to S_k is considered to have a value of zero. Further, we use *basic DTSMP* to refer to the type of DTSMP in which the sojourn time of each state either has a geometric distribution or is deterministic. Note that per its definition, transition probabilities of a basic DTSMP are stationary.

The notion of state-entry probability plays a key role in simplifying model construction and solution for DTSMPs. Consider the very simple DTSMP depicted in Figure 1. In the state diagram, we use a shaded oval that is marked S_k and has a nearby label m_k to denote a state S_k that has a deterministic sojourn time m_k. In addition, an unshaded oval represents a state with geometrically distributed sojourn time. (This convention is used in the remainder of the text.) Hence, for the system shown in Figure 1, S_0's sojourn time has a geometric distribution, while the sojourn time of S_1 is deterministic and is quantified in cycles, namely m_1. We can thus formulate state-entry probabilities by deriving recurrence relations:

$$\begin{aligned} \widehat{P}_0[i] &= \widehat{P}_1[i-m_1] \\ \widehat{P}_1[i] &= \sum_{n=0}^{i-1} p_{01} {p_{00}}^{i-n-1} \widehat{P}_0[n] \end{aligned}$$

The first expression is derived based on the observation that $\mathcal{M}$ will enter S_0 in cycle i if and only if it entered S_1 exactly m_1 cycles ago, as S_1 has a deterministic sojourn time m_1. The second expression must hold since we need to consider all the mutually exclusive paths that lead $\mathcal{M}$ to the most recent transition to S_0 prior to the transition to S_1.

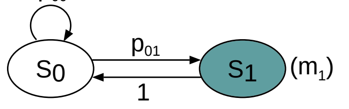


Figure 1. A Simple DTSMP

Hence we let 1) m_j denote the deterministic sojourn time of S_j, 2) p_{jk} denote the probability of the state-transition from S_j to S_k, and 3) Π_d and Π_g denote the subsets of state space Π (such that $\Pi_d \cup \Pi_g = \Pi$) for the states that have deterministic and geometrically distributed sojourn times, respectively. We can then derive the general recurrence relation for a state-entry probability as follows:

Theorem 1 *In a basic DTSMP, state-entry probability* $\widehat{P}_k[i]$ *can be expressed solely in terms of state-entry probabilities as follows:*

$$\widehat{P}_k[i] = \sum_{j \in \Pi_d - \{k\}} p_{jk}\, \widehat{P}_j[i-m_j] + \sum_{j \in \Pi_g - \{k\}} \sum_{n=0}^{i-1} p_{jk}\, {p_{jj}}^{i-n-1} \widehat{P}_j[n] \qquad (1)$$

Proof. The criterion for the correctness of Eq. (1) is that it must exhaustively enumerate all the mutually exclusive paths through which $\mathcal{M}$ will enter S_k from S_j in the ith cycle. Since the sojourn time of each state either has a geometric distribution or is deterministic, the two terms satisfy the criterion at the top level, as they cover the transitions to S_k from the origin states of both types.

In addition, the first term of Eq. (1) exhaustively enumerates the mutually exclusive paths for which the origin state has a deterministic sojourn time, because if a transition to S_j occurs in cycle $i - m_j + n,\ \ 0 < n \leq i - m_j$, then the transition $S_j \rightarrow S_k$ will not happen in cycle i. Conversely, if a transition to S_j occurs in cycle $i - m_j - n$, $0 < n \leq i - m_j$, then a transition $S_j \rightarrow S_x$, $x \neq j$, must occur prior to cycle i. Yet another transition must occur to bring $\mathcal{M}$ back to S_j in the $(i - m_j)$th cycle. We note that this path is already covered by the first term of Eq. (1) such that $\mathcal{M}$ will enter S_k in cycle i.

Furthermore, because $P_j[i] = \sum_{n=0}^{i} {p_{jj}}^{i-n} \widehat{P}_j[n]$, the second term in Eq. (1) can then be rewritten as $\sum_{j\in\Pi_g-\{k\}} p_{jk} \sum_{n=0}^{i-1} {p_{jj}}^{i-n-1} \widehat{P}_j[n] = \sum_{j\in\Pi_g-\{k\}} p_{jk} P_j[i-1]$. Then it is clear that the second term exhaustively covers the mutually exclusive paths via which $\mathcal{M}$ enters S_k in cycle i (from the states that have geometrically distributed sojourn times). Q.E.D.

From Eq. (1), it follows that $\widehat{P}_k[i]$ can be solved and quantified when state-entry probabilities for all the states are initialized. Moreover, Eq. (1) reveals the essence of our approach. In particular, when a recurrence relation involves a transition for which the origin state S_j has a deterministic sojourn time, we let the recursive function for S_j take multiple steps backward in a single (computation) iteration to skip all the intermediate points that constitute the deterministic sojourn time and thus lack the memoryless property. How state-entry probabilities are utilized for the solution of an RRB reward model is described in Section 3.2.

3.2. Utility of State-Entry Probability

3.2.1 Deriving State-Occupancy Probability

As shown below, the notion of state-entry probability allows a state-occupancy probability to have a simple expression.

Theorem 2 *In a basic DTSMP, state-occupancy probability* $P_k[i]$ *can be expressed solely in terms of state-entry* *probabilities as follows:*

$$P_k[i] = \begin{cases} \displaystyle\sum_{n=(i-m_k)+1}^{i} \widehat{P}_k[n] & \textit{if } S_k \textit{ has a deterministic sojourn time} \\ \displaystyle\sum_{n=0}^{i} p_{kk}{}^{i-n}\, \widehat{P}_k[n] & \textit{otherwise} \end{cases} \tag{2}$$

Proof. When the destination state S_k has a deterministic sojourn time, the different values of n mean that $\mathcal{M}$ enters S_k at different points in the range of $[(i-m_k)+1, i]$. Thus, the paths to S_k that are covered by the recurrence relation are mutually exclusive. Furthermore, the lower-bound of the summation, namely $(i-m_k)+1$, ensures that the path coverage is exhaustive. The reason is that a smaller index value will make $\mathcal{M}$ depart from S_k by cycle i.

For the case in which the sojourn time of S_k has a geometric distribution, the value range of n, $[0, i]$, clearly guarantees that the expression $\sum_{n=0}^{i} p_{kk}{}^{i-n}\, \widehat{P}_k[n]$ exhaustively enumerates those mutually exclusive paths through which $\mathcal{M}$ will reside in S_k in cycle i. Q.E.D.

3.2.2 Formulating Expected Accumulated Reward

In order to compute expected accumulated reward, we define the following notation:

$\widehat{W}_k[i]$: A recurrence relation that yields a value equal to the product of the probability that $\mathcal{M}$ will enter S_k at cycle i and the expected value of the reward accumulated up to the end of cycle i conditioned to $\mathcal{M}$ entering S_k at cycle i.

$W_k[i]$: A recurrence relation that yields a value equal to the product of the probability that $\mathcal{M}$ will be in S_k at cycle i and the expected value of the reward accumulated up to the end of cycle i conditioned to $\mathcal{M}$ being in S_k at cycle i.

Our objective is to derive recurrence relations to solve the expected reward accumulated through a mission-task period T which is quantified in number of cycles. Letting this performability measure be denoted as $W[T]$, then by the theorem of total expectation [9], we have

$$W[T] = \sum_{k \in \Pi} W_k[T].$$

The derivation of the recurrence relation for $\widehat{W}_k[i]$ resembles that for $\widehat{P}_k[i]$ in the sense that we divide the origin states (that are involved in the transition to the destination state S_k) into two categories, namely those having deterministic sojourn times and those having geometrically distributed sojourn times. Nonetheless the formulation of $\widehat{W}_k[i]$ must ensure that reward impulse(s) will be appropriately accrued at each point of state transition.

By definition, i must be the point for the occurrence of the transition $S_j \to S_k, j \in \Pi - \{k\}$. Hence we let reward impulses be collected from point n, $n < i$, at which $\mathcal{M}$ enters S_j, up to i at which $\mathcal{M}$ enters S_k. Accordingly, we have the following recurrence relation for $\widehat{W}_k[i]$:

$$\widehat{W}_k[i] = \sum_{j \in \Pi_d - \{k\}} p_{jk} \left(\widehat{W}_j[i-m_j] + \widehat{P}_j[i-m_j]\, m_j\, w_j \right) + \sum_{j \in \Pi_g - \{k\}} \sum_{n=0}^{i-1} p_{jk}\, p_{jj}{}^{i-n-1} \left(\widehat{W}_j[n] + \widehat{P}_j[n](i-n)\, w_j \right) \tag{3}$$

where m_j is the deterministic sojourn time for S_j, and w_j is the magnitude of the reward impulse accrued in each cycle in which $\mathcal{M}$ stays in S_j.

Proof. We first show the correctness of the first term of Eq. (3), which is the expression for the case in which the origin state of a transition to S_k has a deterministic sojourn time. Note that $m_j w_j$ is the reward accrued from the point $(i-m_j)$ to point i, given that $\mathcal{M}$ enters S_j at $i-m_j$ and enters S_k from S_j at i. Then $\widehat{P}_j[i-m_j]m_j w_j$ is the expected value of the reward accrued from $i-m_j$ to i. Then, $\widehat{W}_j[i-m_j] + \widehat{P}_j[i-m_j]m_j w_j$ is the expected value of the reward accrued from 0 to i, given that the transition $S_j \to S_k$ occurs at i. It follows that the expected value of the reward accumulated from 0 to i at which $\mathcal{M}$ enters S_k from S_j equals $p_{jk}\left(\widehat{W}_j[i-m_j] + \widehat{P}_j[i-m_j]m_j w_j\right)$. So when we sum up the above result over $j, j \in \Pi_d - \{k\}$, we exhaustively take into account the mutually exclusive paths that lead $\mathcal{M}$ to enter S_k at i for the case in which the transition's origin state S_j has a deterministic sojourn time. In a similar manner, we can prove the correctness of the second term of Eq. (3). Q.E.D.

To this end we can derive $W_k[i]$ in terms of $\widehat{W}_k[i]$ and $\widehat{P}_k[i]$ as follows:

$$W_k[i] = \begin{cases} \displaystyle\sum_{n=(i-m_k)+1}^{i} \left(\widehat{W}_k[n] + \widehat{P}_k[n](i-n)\, w_k \right) & \text{if } S_k \text{ has a deterministic sojourn time} \\ \displaystyle\sum_{n=0}^{i} p_{kk}{}^{i-n} \left(\widehat{W}_k[n] + \widehat{P}_k[n](i-n)\, w_k \right) & \text{otherwise} \end{cases} \tag{4}$$

Proof. We first discuss the case in which S_k has a deterministic sojourn time. By definition, $\widehat{W}_k[n]$ is the expected value of reward accrued up to cycle n in which $\mathcal{M}$ enters S_k. Further, $(i-n)w_k$ is the additional reward accumulated since cycle n up to cycle i, given that $\mathcal{M}$ enters S_k in cycle n. Then $\widehat{P}_k[n](i-n)w_k$ is the expected value of the additional reward accrued from the state-entry point n to point i. It follows that $\widehat{W}_k[n] + \widehat{P}_k[n](i-n)\, w_k$ is the expected value of the reward accumulated from 0, through n at which $\mathcal{M}$ enters S_k, up to i at which $\mathcal{M}$ remains in S_k. But because S_k has a deterministic sojourn time m_k, the elapsed time since cycle n at which $\mathcal{M}$ enters S_k must be shorter than

m_k; otherwise, $\mathcal{M}$ will depart from S_k by cycle i. Therefore the lower and upper bounds of the summation, namely $(i - m_k) + 1$ and i, respectively, ensure that the summation terms together exhaustively cover all the mutually exclusive paths with which $\mathcal{M}$ will remain in S_k in cycle i.

For the case in which the sojourn time of S_k is geometrically distributed, $\widehat{W}_k[n] + \widehat{P}_k[n](i-n)\, w_k$ is clearly the expected value of the reward accumulated from 0, through n at which $\mathcal{M}$ enters S_k, up to i, given that $\mathcal{M}$ remains in S_k in cycle i. Then ${p_{kk}}^{i-n}\left(\widehat{W}_k[n] + \widehat{P}_k[n](i-n)\, w_k\right)$ is the expected value of the reward accumulated up to i at which $\mathcal{M}$ resides in S_k.

Finally, the summation with the lower and upper bounds of n, namely zero and i, respectively, ensures that the expected values of the accumulated reward that results from the mutually exclusive paths (through which $\mathcal{M}$ enters S_k in cycle n, $n \in \{0, 1, \cdots, i\}$, and remains in S_k in cycle i) are exhaustively taken into account. Q.E.D.

3.2.3 Beyond Basic DTSMP: Periodic Events

Thus far we have derived general formulas for state-entry and state-occupancy probabilities and expressions concerning accumulated reward for basic DTSMPs. Nonetheless, the occurrences of some events in embedded systems are periodic and driven by a countdown timer. Furthermore, such a timer is usually reset upon the completion of the periodic event driven by the timer or upon the completion of another related event in the system, resulting in nonstationary transition probabilities.

In this section, we discuss some concepts concerning such periodic events. Further, we show how state-entry probabilities enable us to solve a model that involves nonstationary transition probabilities associated with timers. We begin by introducing the term *mixed geometric-deterministic sojourn-time distribution*:

Definition 2 *A "mixed geometric-deterministic sojourn-time distribution" (MGDD) is a distribution in which the sojourn time of a state is geometrically distributed and is bounded above by a value L.*

This definition then enables us to define a *simple timer*:

Definition 3 *A "simple timer" is a timer that governs a periodic event $S_j \rightarrow S_i$ in a DTSMP; the sojourn-time distribution of S_j is an MGDD that is characterized by an upper bound L such that when $l < L$ (in which l is the elapsed time since $\mathcal{M}$ enters S_j),*

$$p_{jk} = \begin{cases} 1 & \textit{if } k = j \\ 0 & \textit{otherwise} \end{cases}$$

but when $l = L$,

$$p_{jk} = \begin{cases} 1 & \textit{if } k = i \\ 0 & \textit{otherwise} \end{cases}$$

where $i \in \Pi - \{j\}$.

Accordingly, S_i represents a strictly periodic event (with a period L) which $\mathcal{M}$ undergoes via the transition $S_j \rightarrow S_i$. Indeed, if we add a simple timer to a basic DTSMP, it will remain a basic DTSMP, since a simple timer is defined upon a special case of MGDD for which the upper bound of the sojourn time of S_j can be translated into a deterministic sojourn time for that state (so that in the resulting model each state has either a geometrically distributed or a deterministic sojourn time). On the other hand, the notion of a simple timer is inadequate for modeling periodic events that are more sophisticated. For example, a simple timer will not allow us to model a conditional periodic event. Hence we introduce the concept of *generalized timer* below.

Definition 4 *A "generalized timer" is a timer that governs a periodic event $S_j \rightarrow S_i$ in a DTSMP; the sojourn-time distribution of S_j is an MGDD that is characterized by an upper bound L such that when $l < L$ (in which l is the elapsed time since $\mathcal{M}$ enters S_j),*

$$p_{jk} = \begin{cases} a_k & \textit{if } k \in \Pi - \{i\} \\ 0 & \textit{otherwise} \end{cases}$$

where a_k in $[0, 1)$ if $k \in \Pi - \{i, j\}$, and a_k in $(0, 1]$ if $k = j$ such that $\sum_{k\in\Pi-\{i\}} a_k = 1$. But when $l = L$,

$$p_{jk} = \begin{cases} b_k & \textit{if } k \in \Pi - \{j\} \\ 0 & \textit{otherwise} \end{cases}$$

where b_k in $[0, 1)$ if $k \in \Pi - \{i, j\}$, and b_k in $(0, 1]$ if $k = i$ such that $\sum_{k\in\Pi-\{j\}} b_k = 1$.

It follows that a simple timer is a special case of generalized timer in which $a_k = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{otherwise} \end{cases}$ and $b_k = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{otherwise.} \end{cases}$ Note that Definitions 3 and 4 both imply that a timer is reset and starts countdown upon $\mathcal{M}$'s return to the state (S_j) via which $\mathcal{M}$ underwent a particular periodic event (S_i) upon timer expiration. Moreover, together with the definition of MGDD, Definitions 3 and 4 imply that a timer will be deactivated upon $\mathcal{M}$'s departure from S_j, as the upper bound of the sojourn time of S_j is specified with respect to a continuous occupancy of S_j. Further, since in a DTSMP, $\mathcal{M}$ occupies one and only one state at a time, at any point there will exist at most one activated timer in a DTSMP.

The example illustrated in Figure 2 involves a generalized timer that drives a periodic event $S_0 \rightarrow S_1$. More specifically, the timer is reset to L every time the system enters S_0 (from S_1 or S_2) whose sojourn time has an MGDD. Prior to timer expiration, $p_{01} = 0$ while p_{00} and p_{02} are in $(0, 1)$ such that $p_{00} + p_{02} = 1$. Upon timer expiration, p_{00} becomes 0, whereas the value of p_{02} remains the same. Thus $p_{01} = p_{00}$ such that $p_{01} = 1 - p_{02}$, implying that the system will undergo the periodic event given that $S_0 \rightarrow S_2$ does not happen.

Such a conditional periodic event can be interpreted as follows. A preventive maintenance (S_1) is conducted periodically while the system is in a normal-performance state

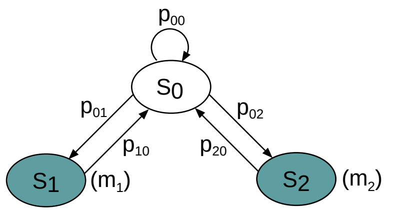


Figure 2. Periodic Event

(S_0). Meanwhile, an unscheduled corrective maintenance (S_2) will override the preventive-maintenance schedule if an error is detected in the system. Then, the point at which a maintenance (of either type) completes will mark an epoch for the interval L preceding the next preventive maintenance event. Moreover, we note that internal-state changes that are not covered by monitoring mechanisms and that do not result in appreciable damage to the system are often transparent to the application. This suggests that when a system is in a state subset that represents a series of performance degradations, a scheduled maintenance will occur as usual until the extent of the degradation becomes excessive (e.g., resulting in an exception or failure). Therefore, in order to build and solve RRB models for systems involving more sophisticated periodic events, we define the notion of *degradation (or progress) sequence* below.

Definition 5 *A "degradation (or progress) sequence" is an ordered random sequence that is formed by a state subset $J = \{j_k \mid k \in \{0, \cdots, |J|-1\}\}$ where the sojourn time of each state j_k has a geometric distribution such that the value of $p_{j_k j_{k'}}$ is in $(0, 1)$ if $k' - k \in \{0, 1\}$ and is zero otherwise.*

Suppose a timer is reset every time $\mathcal{M}$ enters S_{j_0}. Then, while $\mathcal{M}$ experiences successive internal-state changes, the timer countdown will continue; upon timer expiration, $\mathcal{M}$ will go from state j_k (in which $\mathcal{M}$ resides when the timer expires) to a periodic event. In turn, that suggests to us the concept of a *collective sojourn time*:

Definition 6 *A "collective sojourn time" l_{j_n} is the accumulated time during which $\mathcal{M}$ continuously stays in a degradation sequence with a state subset J, via the path $S_{j_0} \rightarrow S_{j_1} \rightarrow \cdots \rightarrow S_{j_{n-1}} \rightarrow S_{j_n}$, where $n \leq |J| - 1$.*

Coupled with Definition 5, Definition 6 yields the following theorem:

Theorem 3 *If a degradation sequence with a state subset J has an upper bound L_J on its collective sojourn time, then the sojourn-time distribution of each state in J is an MGDD.*

Proof. Per Definition 5, the sojourn time of each state in a degradation sequence is geometrically distributed. Since the collective sojourn time for the degradation sequence is bounded above by L_J, the sojourn time of S_{j_k}, $j_k \in J$, will have an upper bound $L_J - l_{k-1}$, where the value of l_{k-1} is in $(0, L_J)$ if $k \geq 1$ and is zero otherwise. Q.E.D.

Together with the concepts developed above, the notion of state-entry probability enables us to model systems involving a generalized timer and a degradation (or progress) sequence. More specifically, by carefully determining the index of a state-entry probability and the boundary points, we are able to adapt the general formulas (for basic DTSMP) developed in Sections 3.1 and 3.2 to precisely represent systems with a generalized timer. In the following we discuss our approach to adapting those general formulas. A more comprehensive example application is elaborated in the next section.

Let us revisit the state diagram in Figure 2, where S_1 represents a periodic event to which the system normally goes from S_0 every L cycles. (Without losing generality, we can state that the system has a degradation sequence that is degenerate, as $J = \{j_0\}$ and $j_0 = 0$.) In addition, S_2 represents an unscheduled event. As mentioned earlier, the generalized timer will reset to L upon the completion of either event. Hence we have,

$$\widehat{P}_0[i] = \widehat{P}_1[i - m_1] + \widehat{P}_2[i - m_2] \qquad (5)$$

$$\widehat{P}_1[i] = {p_{00}}^{L-1}\, \widehat{P}_0[i - L] \qquad (6)$$

$$\widehat{P}_2[i] = \sum_{n=(i-L)+1}^{i-1} p_{02}\, {p_{00}}^{i-n-1} \widehat{P}_0[n] \qquad (7)$$

Clearly Eq. (5) is a direct application of the first term in Eq. (1) with $p_{10} = 1$ and $p_{20} = 1$. Note also that Eq. (6) is the result of an application of the second term in Eq. (1), as the sojourn time of S_0 is geometrically distributed prior to $\mathcal{M}$'s departure to the periodic event represented by S_1. Then, when we contrast the state diagram with Eq. (1), we note that 1) zero is the only valid value for j (i.e., S_0 is the only origin state for a transition to S_1), 2) the periodic event will occur if and only if $\mathcal{M}$ has stayed in S_0 since the point $i - L$, and 3) $p_{01} = 0$ until the ith cycle. Hence such an application of Eq. (1) allows us to obtain Eq. (6) in the following manner:

$$\begin{aligned} \widehat{P}_1[i] &= \sum_{j \in \Pi_g - \{1\}} \sum_{n=0}^{i-1} p_{j1} p_{jj}^{i-n-1} \widehat{P}_j[n] \\ &= p_{01} {p_{00}}^{i-(i-L)-1}\, \widehat{P}_0[i - L] \\ &= {p_{00}}^{L-1}\, \widehat{P}_0[i - L] \end{aligned}$$

Further, as S_0 is the only origin state for the transition to S_2, by applying Eq. (1) again, we are able to derive the solution of $\widehat{P}_2[i]$ (i.e., Eq. (7)) as follows.

$$\begin{aligned} \widehat{P}_2[i] &= \sum_{j \in \Pi_g - \{2\}} \sum_{n=0}^{i-1} p_{j2} p_{jj}^{i-n-1} \widehat{P}_j[n] \\ &= \sum_{n=(i-L)+1}^{i-1} p_{02} p_{00}^{i-n-1} \widehat{P}_0[n] \end{aligned}$$

It can be shown that because of the underlying recurrence relations, $(i-L)+1$ is the earliest point that should be taken into account by Eq. (7). We omit the proof here due to space limitations.

4. An Example Application

4.1. Image-Based Tracking System

The DTSMP in Figure 3 represents an image-based tracking system that is typically embedded in an autonomous vehicle. The system's mission task is to track a moving target by image-taking and predicting the target's subsequent position according to its trajectory. The trajectory is updated per the most recent image. Toward the end of the $(i-1)$th iteration, the camera takes yaw and pitch adjustments per the prediction, followed by another image-taking; then the system goes to the ith iteration. Clearly this is a typical embedded-system application that executes in a closed loop. Note that such a closed-loop computation may allow undetected errors to manifest and accumulate to cause target missing (i.e., S_4 in Figure 3) or inaccurate camera positioning (i.e., S_3 in Figure 3). Also, the accumulated inaccuracy may eventually become excessive and lead to target missing. In addition, because the embedded software involves image processing and pattern recognition, the system is vulnerable to memory exhaustion.

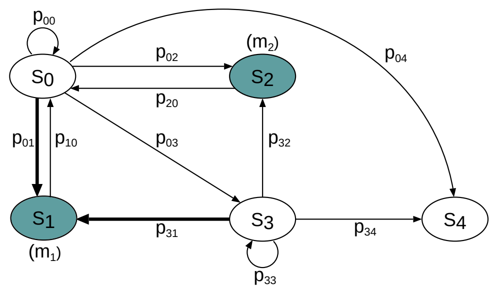


Figure 3. Image-Based Tracking System

Accordingly, proactive restart (i.e., S_1) is applied every L cycles to let the system regain adequate capacity (i.e., going back to S_0). Moreover, in order to reduce the performance cost of the preventive measure, a policy is enforced such that the proactive-restart timer will be reset immediately after a reactive restart (i.e., S_2, which involves a system reboot and re-initialization) caused by a detected error. Table 1 summarizes the state definitions.

Table 1. State Definitions

State	Definition
S_0	Normal system performance.
S_1	Periodic proactive restart.
S_2	Reactive restart.
S_3	Reduced accuracy caused by undetected errors.
S_4	Missing of target.

Apparently, S_0 and S_3 constitute a degradation sequence in which $j_0 = 0$ and $j_1 = 3$. Moreover, the degradation sequence has an upper bound for its collective sojourn time equal to the inter-restart interval L. It follows that transitions to S_1 will be driven by a generalized timer. Further, the timer is reset to L every time the system enters S_0.

4.2. Analytic Solution

In the following subsections, we discuss one or a few equations for each type of probabilistic measure rather than explain all the equations due to space limitations.

4.2.1 Solution of State-Entry Probability

First, we note that both of the origin states for the transitions into S_0 have deterministic sojourn times and that S_0 itself does not represent a periodic event. So the recurrence relation for $\widehat{P}_0[i]$ is a straightforward application of Eq. (1):

$$\widehat{P}_0[i] = \widehat{P}_1[i-m_1] + \widehat{P}_2[i-m_2] \tag{8}$$

However, S_1 represents a periodic event, and the system can go to S_1 from either S_0 or S_3. In addition, the timer for the periodic event is reset and starts counting down upon the completion of the event (marked by the most recent entry to S_0). Hence,

$$\widehat{P}_1[i] = {p_{00}}^L \widehat{P}_0[i-L] + \sum_{n=0}^{L-1} {p_{33}}^{L-n-1} p_{03}\, {p_{00}}^n \widehat{P}_0[i-L] \tag{9}$$

Eq. (9) thus captures the system behavior influenced by the relationship between two different types of system restart by backtracking to the most recent entry to S_0 (which implies the completion of the most recent system restart, proactive or reactive).

Next, because S_3 can only be reached from S_0 and because the system undergoes a periodic proactive restart when the elapsed time since the most recent entry to S_0 is L, we have the following equation for $\widehat{P}_3[i]$,

$$\widehat{P}_3[i] = \sum_{n=(i-L)+1}^{i-1} p_{03}\, {p_{00}}^{i-n-1} \widehat{P}_0[n] \tag{10}$$

Note that Eq. (10) is analogous to Eq. (7). The reason is that, although semantically different, both S_3 and S_2 (respectively) in Figures 3 and 2 are a destination state of a transition from a state through which the system can go to a periodic event.

4.2.2 Solution of State-Occupancy Probability

According to Definition 2, the sojourn-time distribution of S_0 is an MGDD. Therefore, while the following equation is generally an application of Eq. (2), the lower bound of the summation is set to $(i-L)+1$ rather than zero.

$$P_0[i] = \sum_{n=(i-L)+1}^{i} {p_{00}}^{i-n} \widehat{P}_0[n] \tag{11}$$

As to $P_4[i]$, since it is an absorbing state (meaning that $p_{44} = 1$), by applying the second term of Eq. (2) we have

$$P_4[i] = \sum_{n=0}^{i} p_{44}{}^{i-n} \widehat{P}_4[n] = \sum_{n=0}^{i} \widehat{P}_4[n] \tag{12}$$

4.2.3 Solution of Expected Accumulated Reward

We first discuss the formulation of $\widehat{W_0}[i]$. Because the origin states for a transition to S_0, namely S_1 and S_2, both have deterministic sojourn times, we apply the first term of Eq. (3) so that we have

$$\widehat{W_0}[i] = \sum_{j\in\{1,2\}} \left(\widehat{W_j}[i-m_j] + \widehat{P}_j[i-m_j]m_j w_j\right) \tag{13}$$

where p_{j0} does not appear as both p_{10} and p_{20} equal 1.

The formulation of $W_0[i]$ is an application of Eq. (4):

$$W_0[i] = \sum_{n=(i-L)+1}^{i} p_{00}{}^{i-n} \left(\widehat{W_0}[n] + \widehat{P}_0[n](i-n)w_0\right) \tag{14}$$

We note that the boundary point $(i - L) + 1$ is again used for the summation in the above equation because the sojourn-time distribution of S_0 is an MGDD.

Further, since S_4 is an absorbing state that represents the loss of a mission task, entering S_4 will negate all the reward accumulated prior to the task loss. Hence $\widehat{W_4}[i] = 0$. Moreover, reward accrual will cease upon entering S_4. So $w_4 = 0$. Then applying Eq. (4) yields

$$W_4[i] = \sum_{n=0}^{i} p_{44}{}^{i-n} \left(\widehat{W_4}[n] + \widehat{P}_4[n](i-n)\, w_4\right) = 0 \tag{15}$$

4.3. Evaluation Results

Applying the recurrence relations and using Mathematica, we solve a number of probabilistic measures for the image-based tracking system. In the first study, we evaluate the state-occupancy probabilities, $P_0[k]$ and $P_1[k]$, the probabilities that the system is in normal operation and is undergoing proactive restart in cycle k, respectively. The parameter value assignment is shown in Table 2.

Recall that the value of w_k is the magnitude of the reward impulse for a cycle during which the system resides in S_k, and m_k is the deterministic sojourn time of S_k. In addition, q is the probability of the occurrence of a nonfatal but accumulative error, and c is the coverage of error detection.

Note also that the magnitude of the reward impulse for S_1 is set to 0.2, because during such a restart, the system remembers the target velocity and trajectory known from the most recent calculation, so that once the system is done with the proactive restart, it will be able to predict the target location in order to adjust the camera's position. Thus the costly re-initialization that is required by a reactive restart is avoided. Further, a reward impulse with a magnitude of 0.75 is assigned to S_3, the state in which there exists an error in camera positioning because of the system's inability to accurately predict the target location.

Table 2. Parameter Values

Parameter	*Value*	*Parameter*	*Value*
p_{00}	$1 - p_{02} - p_{03} - p_{04}$	c	0.80
p_{02}	cq	w_0	1
p_{03}	$(1-c)q$	w_1	0.2
p_{04}	0.00001	w_2	0
p_{32}	0.01	w_3	0.75
p_{33}	$1 - p_{32} - p_{34}$	w_4	0
p_{34}	0.01	m_1	5
q	0.005	m_2	10

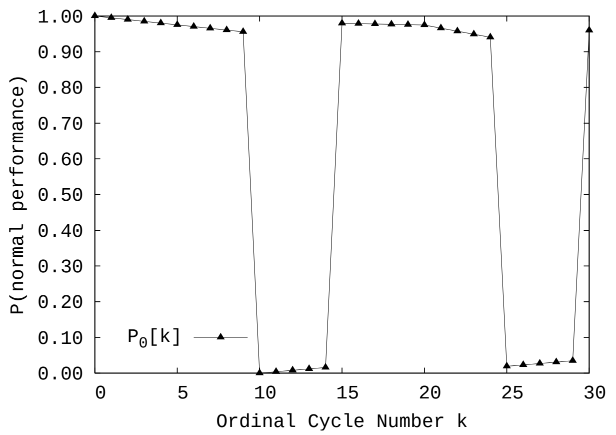
Throughout this application, we assume that 1,000 cycles are allocated to a mission task of the image-based tracking system. But for clarity, in Figures 4(a) and 4(b) we show only the results of $P_0[k]$ and $P_1[k]$ for the first 30 cycles. In addition, for the sake of illustration (i.e., to show multiple "spates" in a short period of time), we let the interval between two consecutive system restarts, L, be 10 cycles.

From Figures 4(a) and 4(b), we observe that the two state-occupancy probabilities approximately complement each other. That is a reasonable result because when it is highly probable (see the explanation for why this probability is less than 1 shortly) that the system is engaged in a proactive restart (i.e., in S_1), the probability that the system is under normal operation must be low, and vice versa.

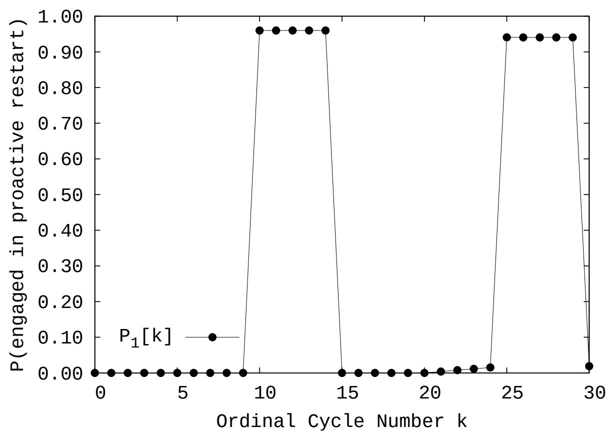
In addition, we can observe from 4(a) that starting from the initial point (i.e., $k = 0$), the value of $P_0[k]$ is continuously decreasing; nonetheless upon the completion of a proactive restart (i.e., $k = 15$ and $k = 30$), the value of $P_0[k]$ appreciably rebounds. On the other hand, the value prior to the second proactive-restart (at $k = 25$) is less than the value right before the first restart (at $k = 10$), and so are their post-restart values (at $k = 15$ and $k = 30$). This suggests that proactive restart is a means of slowing down, rather than eliminating, a system's capacity deterioration.

Another interesting phenomenon is that the values of $P_0[k]$ are greater than zero in the interval during which the system is expected to be engaged in proactive restart (i.e., between cycles 10 and 15, and between cycles 25 and 30). The reason is that the system might undergo a reactive restart due to a detected error prior to a scheduled point for proactive restart, which results in a timer reset so that the system will not go to proactive restart until the elapsed time since the reset reaches L.

The model also allows us to evaluate the image-tracking system's reliability and normal-performance probability at the end of the task period, as shown in Figures 5(a) and 5(b). Specifically, $1 - P_4[k]$ is regarded as the system's reliability and $P_0[k]$ is the probability of normal performance. Figures

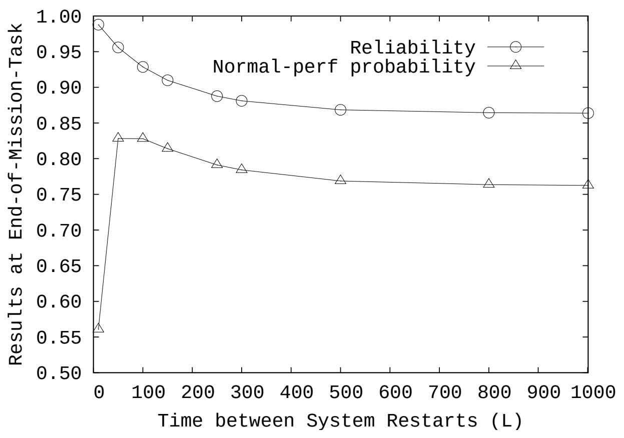


(a) $P_0[k]$

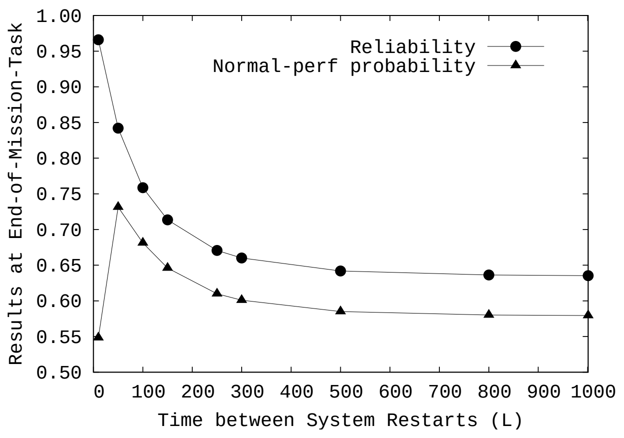


(b) $P_1[k]$

Figure 4. State-Occupancy Probabilities



(a) Low Failure Probability Case



(b) High Failure Probability Case

Figure 5. Probability Measures

5(a) and 5(b) illustrate the cases in which failure probability p_{34} equals 0.002 and 0.01, respectively. The time interval between two system restarts, L, is kept as a variable while the measures are evaluated at the points where L equals 10, 50, 100, 150, 250, 300, 500, 800, and 1001. Note that $L = 1001$ implies that proactive restart is never applied during the mission task period of 1,000 cycles.

As shown by the curves, in the high-failure-probability case, there is a clear maximum of the probability that the system will be in the normal-performance mode, which occurs when $L = 50$ and implies an optimal restart frequency of $\frac{1}{L}$. However, for the low-failure-probability case, the maximum is less clear. In particular, the value of the normal-performance probability drops slowly as L increases, suggesting that proactive restart offers only limited benefit to a highly reliable system. Similarly, the curves in Figures 5(a) and 5(b) imply that the worth of proactive restart with respect to reliability improvement for the high-failure-probability case is far more significant than that for the low-failure-probability case (i.e., 52% versus 14% when $L = 10$, and 33% versus 10% when $L = 50$).

Nonetheless we understand that reliability is not the sole criterion for determining the frequency of proactive restart. In particular, while a high value of $\frac{1}{L}$ greatly improves reliability, the performance cost of frequent proactive restarts may negate much of the benefit, so that the value of normal-performance probability decreases, as illustrated by the curves in Figures 5(a) and 5(b). As a result, the two separate measures are unable to help identify a value of L that is the best for the accomplishment of a mission task.

Therefore, in the subsequent study, we solve a performability measure, namely mission accomplishment evaluated as the expected accumulated reward $W[T]$. As illustrated by Figure 6, the quantitative results for $W[T]$ imply that the optimal intervals between consecutive system restarts are different for the low- and high-failure probability cases. More specifically, the optimal values of L for the former and latter cases are 100 and 50, respectively.

In addition, we observe that in the high failure probability case there exists a sharp peak resulting from the optimal L, while the other case lacks such a clear peak. We can use the results of reliability and normal-performance probability (shown in Figures 5(a) and 5(b)) to interpret the optimal values of L. More specifically, the tradeoffs between the

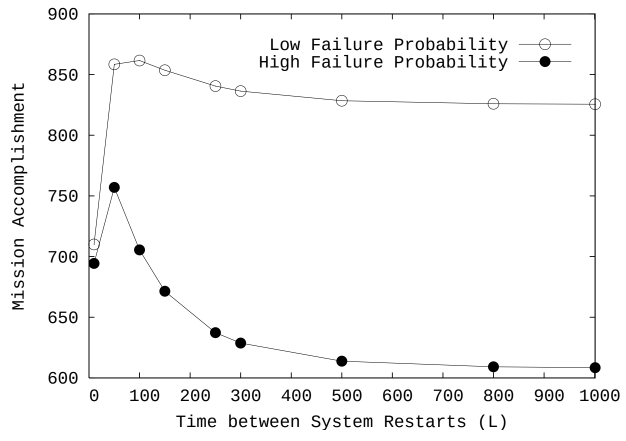


Figure 6. Expected Accumulated Reward

reliability improvement from and the performance cost of system restart determine the peak locations and the sharpness of the peaks.

By setting the magnitudes of reward impulses differently and computing time-averaged accumulated reward, we have also evaluated interval availability (a measure indeed quantifies a duty cycle) for the image-based tracking system.

5. Concluding Remarks

We have developed a performability evaluation framework for an important class of embedded systems, namely closed-loop applications. Although prior research efforts resulted in a number of formalisms for modeling embedded systems, e.g., timed automaton [10], their primary purpose was system property verification and validation. In contrast, the emphasis of our work is an approach to systematically constructing and analytically solving RRB reward models for quantitative evaluation. Additional reasons why this effort is worthwhile are the following.

First, the RRB approach enables straightforward computation for various measures of nonfunctional system properties, including instant-of-time availability, interval-of-time availability, reliability, expected accumulated reward, and time-averaged accumulated reward, which are meaningful for quantifying gracefully degradable performance of iterative embedded applications.

Moreover, it is noteworthy that expected accumulated reward and time-averaged accumulated reward are conceptually coherent with, respectively, the notion of return and duty cycle for embedded systems. This makes the RRB framework a good match with its target application. Coupled with solution simplicity, this match makes RRB models well-suited for early design assessment and trend study of design alternatives for embedded systems.

In addition, as we did a preliminary study in [8], a hybrid and hierarchical approach will enable an RRB reward model to represent the behavior of more sophisticated embedded systems, such as distributed embedded systems. In particular, we built a two-layer model for the assessment of coordinated software and hardware fault tolerance in a distributed avionics system. While the upper layer was a recurrence-relation-based reward model, the lower-layer model was built upon stochastic activity networks (SANs). By assigning different initial markings to the SAN model, the lower layer was used to evaluate the state transition probabilities required by the upper layer. The upper layer model then used the results to compute the expected accumulated reward. In this manner, we were able to model a continuous-time stochastic process. Therefore, when supported by hybrid/hierarchical methods, applications beyond closed-loop embedded systems may have the potential to leverage the RRB approach. Possible applications are those “interval-of-time” problems in which system behavior within a fixed interval (which we view as a generalized operational cycle) can be summarized by a state transition, for example, checkpointing-interval optimization and clock-synchronization frequency determination.

References

[1] S. Mohanty and V. K. Prasanna, “Design of high-performance embedded system using model integrated computing,” in *Proc. of the 2nd RTAS Workshop on Model-Driven Embedded Systems*, (Toronto, Canada), May 2004.

[2] W. H. Sanders *et al.*, “Multi-formalism and multi-solution-method modeling frameworks: The Möbius approach,” in *Proc. of Symp. on Performance Evaluation – Stories and Perspectives*, (Vienna, Austria), pp. 241–256, Dec. 2003.

[3] F. Stenberg, R. Manca, and D. Silvestrov, “Empirical estimation for discrete-time semi-Markov processes with applications in reliability,” *Journal of Nonparametric Statistics*, vol. 18, pp. 483–498, Oct. 2006.

[4] M. R. Sternberg and G. A. Satten, “Discrete-time nonparametric estimation for semi-Markov models of chain-of-events data subject to interval censoring and truncation,” *Biometrics*, vol. 55, pp. 514–522, June 1999.

[5] S. Mercier, “Numerical bounds for semi-markovian quantities and application to reliability,” *Methodology and Computing in Applied Probability*, July 2007.

[6] E. S. Rieger and G. Hasslinger, “An analytical solution for the discrete time single server system with semi-markovian arrivals,” *Queueing Systems*, vol. 18, pp. 69–105, Mar. 1994.

[7] A. T. Tai, S. N. Chau, and L. Alkalai, “COTS-based fault tolerance in deep space: A case study on IEEE 1394 application,” *International Journal of Reliability, Quality and Safety Engineering*, vol. 9, pp. 17–40, Mar. 2002.

[8] A. T. Tai and W. H. Sanders, “Performability modeling of coordinated software and hardware fault tolerance,” in *Proc. of PMCCS-6*, (Monticello, Illinois), pp. 31–34, Sept. 2003.

[9] K. S. Trivedi, *Probability & Statistics with Reliability, Queueing, and Computer Science Applications*. New York: John Wiley & Sons, 2nd ed., 2002.

[10] N. A. Lynch and F. W. Vaandrager, “Action transducers and timed automata,” *Formal Aspects of Computing*, vol. 8, no. 5, pp. 499–538, 1996.