Development Programs for One-Shot Systems Using Multiple-State Design Reliability Models

Suntichai Shevasuthisilp
Chiang Mai University

Stephen B. Vardeman
Iowa State University, vardeman@iastate.edu

Follow this and additional works at: http://lib.dr.iastate.edu/stat_las_preprints

Part of the Statistics and Probability Commons

Recommended Citation
http://lib.dr.iastate.edu/stat_las_preprints/105

This Article is brought to you for free and open access by the Statistics at Iowa State University Digital Repository. It has been accepted for inclusion in Statistics Preprints by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Development Programs for One-Shot Systems Using Multiple-State Design Reliability Models

Abstract
Design reliability at the beginning of a product development program is typically low and development costs can account for a large proportion of total product cost. We consider how to conduct development programs (series of tests and redesigns) for one-shot systems (which are destroyed at first use or during testing). In rough terms, our aim is to both achieve high final design reliability and spend as little of a fixed budget as possible on development. We employ multiple-state reliability models. Dynamic programming is used to identify a best test-and-redesign strategy and is shown to be presently computationally feasible for at least 5-state models. Our analysis is flexible enough to allow for the accelerated stress testing needed in the case of ultra-high reliability requirements, where testing otherwise provides little information on design reliability change.

Keywords
development programs, one-shot systems, multiple-state design reliability, test, redesign, optimal programs, dynamic programming, accelerated testing

Disciplines
Statistics and Probability

Comments
Abstract: Design reliability at the beginning of a product development program is typically low and development costs can account for a large proportion of total product cost. We consider how to conduct development programs (series of tests and redesigns) for one-shot systems (which are destroyed at first use or during testing). In rough terms, our aim is to both achieve high final design reliability and spend as little of a fixed budget as possible on development. We employ multiple-state reliability models. Dynamic programming is used to identify a best test-and-redesign strategy and is shown to be presently computationally feasible for at least 5-state models. Our analysis is flexible enough to allow for the accelerated stress testing needed in the case of ultra-high reliability requirements, where testing otherwise provides little information on design reliability change.

Keywords: development programs, one-shot systems, multiple-state design reliability, test, redesign, optimal programs, dynamic programming, accelerated testing

1. INTRODUCTION

The purpose of this article is to identify and study the properties of optimal (test-and-redesign) development programs. We consider programs for one-shot systems such as missiles, which are destroyed at their first test or first use. Our analysis builds on those of Huang, McBeth, and Vardeman [1] (HMV1) and Moon, Vardeman, and McBeth [2] (MVM2). The previous work is generalized in two important directions, by allowing 1) multiple-state reliability modeling and 2) test failure probabilities different from normal-use failure probabilities.

Earlier work (HMV1 and MVM2) provided only crude 2-state modeling of design reliability. That modeling might be appropriate for systems with only a single potential failure mode. But the present multiple-state analysis allows for “fine” modeling of the evolution of a design (where multiple failure modes are possible and might be either eliminated or inadvertently generated by an engineering redesign).

Real test failure probabilities can be different from normal-use-condition failure probabilities in the important case of accelerated stress testing. Such testing can be necessary for sensible development programs for high-reliability systems. There, it is practically impossible to obtain normal-use-condition test results that will definitively signal a change in design reliability. However, using appropriate physical acceleration factors, it can be possible to raise failure probabilities and thereby conduct tests whose results more clearly distinguish between design reliabilities. Feinberg and Gibson [3] and Meeker and Escobar [4] note that accelerated testing has been used extensively in the development of products like semiconductors, microelectronics, lasers, electronic devices, and mechanical components, to obtain timely information on the reliability of both products and components. Breyfogle [5] and Meeker and Escobar [4] note that testing can be accelerated by increasing the use-rate, the aging-rate, or the level of other physical stress on a product.

Our results can provide guidance for development programs for military systems and in some in-house industrial contexts, when there is a fixed budget for development and production. Our method for finding optimal policies is not complicated and is computationally feasible.

2. MODELING DEVELOPMENT PROGRAMS USING MULTIPLE-STATE DESIGN RELIABILITY MODELS
We make the following basic assumptions about the process used in the development of a one-shot system.

1) An initial budget is sufficient to build \( n \) systems and all costs are in units of systems built.
2) Testing does not change design reliability. It provides information on the current design reliability state. A test result is either a “success” or a “failure” and can be purchased at a cost \( t \). (As this test information is Bernoulli distributed, it does not typically accumulate very rapidly.)
3) Redesign has the potential to change the design reliability, but in general does not necessarily always improve it. It might degrade or improve design reliability, and can be purchased at a cost \( d \).
4) Constants \( n, t, d \) are greater than 0 (and are not necessarily integers).
5) The effects of redesign are described by the redesign transition matrix \( u \) (explained more fully below) at any point in the development process where it is employed.
6) There are \( k \) possible values of design (normal-use-condition) reliability (\( k \) design reliability states). For convenience, design reliability at a higher numbered state is greater than design reliability at a lower numbered state \( (r_k > r_{k-1} > \ldots > r_1) \).
7) For each reliability state \( i \), there is a corresponding test reliability, \( q_i \). We will assume that test reliabilities are ordered in the same way as design reliabilities, i.e. \( (q_k > q_{k-1} > \ldots > q_1) \).
8) There is no restriction on the order of activities in a development plan. Multiple redesigns can be performed in a row (in case the current design reliability is thought to be low or testing is expensive). Multiple tests can be performed in a row (in case it is desirable to definitively ascertain the current design reliability).

### 2.1 The General Multiple-State Design Reliability Models

We seek a development program that produces the largest possible mean number of effective systems of a final design, given an initial budget sufficient to build \( n \) systems. We will work in units of “systems” and final (conditional) mean numbers of effective systems can be evaluated from the remaining budget at the end of development as \( B^* \cdot \Pi^* \) (for \( B^* \) the part of the budget remaining at the end of development, \( \cdot \) the greatest integer function, and \( \Pi^* \) the final design reliability). Then \( E \left( \Pi^* \cdot \left\lfloor B^* \right\rfloor \right) \) is our objective function. We will take what amounts to a Bayesian approach to the optimization and let \( V^*_n(\xi) \) be the maximum of this objective function (the overall return of an optimal development plan) given the initial budget of \( n \) systems and the starting probability distribution over the states \( (\xi_0 = \xi) \). This is a problem in sequential analysis. A development process proceeds in stages. At any stage of a development program, there are 3 choices of development activity: “test,” “redesign,” and “build.” Each activity has a different conditional expected pay-off, which we proceed to explain in detail.

#### 2.1.1 Testing

Testing provides information on the current design reliability state by producing a binary test result: a success or a failure on any test. Each test can be purchased at cost of \( t \) systems and a sample size of “one” is used for testing. Bayes’ rule is used to update one’s distribution for the current reliability state after a test is made. This, of course, requires knowledge of reliabilities of the states \( (\xi) \), and a pre-test probability distribution over the states \( (\xi) \). We will let \( \eta(\xi) \) denote a vector specifying the updated probability distribution \( (\xi') \) after testing. The 2 possible versions of \( \eta(\xi) \) based on the test result \( (X) \) are \( \eta_0(\xi) \) (if the test is successful) and \( \eta_1(\xi) \) (if the test is a failure). The forms of \( \eta_0(\xi) \) and \( \eta_1(\xi) \) are generalizations of those obtained in HMV1. Let \( q \) be the vector of test reliabilities, \( r(\xi) \) be the expected reliability under normal use conditions, that is
\[ r(s) = r_1 \cdot s_1 + r_2 \cdot s_2 + \ldots + r_k \cdot s_k \]  
(2.1)

and \( q(s) \) be the expected reliability under test conditions,

\[ q(s) = q_1 \cdot s_1 + q_2 \cdot s_2 + \ldots + q_k \cdot s_k \]  
(2.2)

(For case of testing under normal use conditions, \( q = r \) and \( q(s) = r(s) \).) Then the updated distribution over reliability states following a test is

\[ \tilde{s}_0 \equiv (\eta_{01}(s), \eta_{02}(s), \ldots, \eta_{0k}(s)), \]

if a test is successful, \( X = 0 \)

where

\[ \eta_{0i}(s) = \frac{s_i \cdot q_i}{q(s)} \]

for \( i = 1, 2, \ldots, k \)  
(2.3)

and

\[ \tilde{s}_1 \equiv (\eta_{11}(s), \eta_{12}(s), \ldots, \eta_{1k}(s)), \]

if a test is a failure, \( X = 1 \)

where

\[ \eta_{1i}(s) = \frac{s_i \cdot (1 - q_i)}{1 - q(s)} \]

for \( i = 1, 2, \ldots, k \)  
(2.4)

The remaining budget after testing will be \( n - t \). Therefore, upon testing, the optimal conditional expected numbers of effective systems will be

\[ V_{n-t}(\eta_{00}(s)) \] (if a test is successful)

or

\[ V_{n-t}(\eta_{10}(s)) \] (if a test is a failure).

So the expected final return if a test is made is

\[ q(s) \cdot V_{n-t}(\eta_{00}(s)) + (1 - q(s)) \cdot V_{n-t}(\eta_{10}(s)). \]

2.1.2 Redesigning

We suppose that redesign is purchased at a cost of \( d \) systems lost to a final stockpile per unit of engineering effort expended in attempts to improve current design reliability. Our model allows the possibility of regressive redesigns (degrading design reliability). The effect of a redesign is represented by a (stationary) Markov chain transition matrix \( u \) (see the particular structures of the matrix used in our work in Appendix A.6) describing movements between design reliability states (to better or worse states) as shown in Figure 1.
Let \( \delta (\mathbf{s}) \) be an updated probability distribution over the states \( \mathbf{s} \) produced by a redesign. The form of this is a generalization of the form from MVM2. Each
\[
\delta_i (\mathbf{s}) = s_1 \cdot u_{i1} + s_2 \cdot u_{i2} + \ldots + s_k \cdot u_{ik}
\]
for \( i = 1, 2, \ldots, k \),
or in matrix notation
\[
\mathbf{s'} \equiv (\delta_1 (\mathbf{s}), \delta_2 (\mathbf{s}), \ldots, \delta_k (\mathbf{s})) = (s_1, s_2, \ldots, s_k) \cdot \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & & \vdots \\ u_{k1} & u_{k2} & \cdots & u_{kk} \end{pmatrix}
\]
(2.5)

The remaining budget after a redesign will be \( n - d \). Therefore the optimal expected return if one redesigns is \( V_{n-d} (\mathbf{\delta(s)}) \).

### 2.1.3 Building

The final potential development activity is “build” which means that the development program is terminated and the entire remaining budget is used to build systems according to the current design and with its reliability. Therefore if one builds, the mean number of effective systems in the final stockpile is \( n \cdot r(\mathbf{s}) \).

### 2.1.4 Optimal Return Functions

In light of the foregoing development, the overall optimal return function is
\[
V_n (\mathbf{s}) = \max \{ \Psi_1, \Psi_2, \Psi_3 \}
\]
(2.6)
for
\[
\Psi_1 = [n] \cdot r(\mathbf{s}),
\Psi_2 = q(\mathbf{s}) \cdot V_{n-1} (\mathbf{1}_0 (\mathbf{s})) + (1 - q(\mathbf{s})) \cdot V_{n-1} (\mathbf{1}_1 (\mathbf{s}))
\]
and
\[
\Psi_3 = V_{n-d} (\mathbf{\delta(s)})
\]
A development activity is (currently or initially) optimal at budget \( n \) and probability distribution \( \Psi \) if its corresponding \( \Psi \) is maximum in display (2.6).

Optimal activities at any stage of a development program can be determined by repeatedly updating the remaining budget and probability distribution over the states and using the optimal return function. An optimal development program will continue sequentially until “build” is chosen. In the case that testing and redesign both cost 1 system, the number of possible development policies could be as large as \( \frac{2 \cdot 3^n + 3^{n-1} - 1}{2} \). Naïve direct enumeration of all possible development policies to find a best plan would thus require that one find expected payoffs for each of a set of policies whose size grows exponentially in \( n \).

### 2.2 Accelerated Stress Testing

We consider a particular model for accelerated stress testing made from the general \( k \)-state model. Our test failure probability vector is obtained by multiplying \( p \) by an “acceleration constant.” Therefore the modified test failure probability vector is “bigger” than under normal use conditions. Test reliability under accelerated testing is

\[
q_i = 1 - p_i \cdot a_f \quad \text{for } i = 1, 2, \ldots, k
\]

or

\[
q_i = 1 - (1 - r_i) \cdot a_f
\]

(\( a_f = 1 \) and \( q_i = r_i \) describe normal use conditions and \( 1 < a_f \leq p_i^{-1} \) describes accelerated testing.)

### 3. SOME DIRECT CONSEQUENCES OF THE MODEL ASSUMPTIONS

Despite the move from 2-state models to \( k \)-state models and the generalization that allows \( q \neq r \), many properties of the model specified in Section 2 carry over directly from corresponding properties of the 2-state model of MVM2. For completeness, these model properties are summarized in this section.

#### 3.1 Properties of the Update of \( s \) After a Test

**Proposition 1** The expected design reliability after testing is the same as the expected current design reliability:

\[
q(\xi) \cdot r(\xi_0(\xi)) + (1 - r(\xi)) \cdot q(\xi_1(\xi)) = r(\xi)
\]

This direct generalization of the 2-state Proposition 1 of MVM2 reflects the fact that testing does not change the design reliability. Testing only provides information on current design reliability.

**Proposition 2** (Probability distribution over the states \( (s_1, s_2, \ldots, s_k) \) under an infinite sequence of tests) If one could make infinite series of tests on particular design, \( s_0 \) would converge in probability to a distribution degenerate at the correct reliability state.

This generalizes Proposition 2 of MVM2 and confirms that our modeling allows that with enough testing, one could with virtual certainty ascertain the true current design reliability.

**Proposition 3** \( \eta_{1k}(\xi) \leq s_k \leq \eta_{0k}(\xi) \) and \( \eta_{01}(\xi) \leq s_1 \leq \eta_{11}(\xi) \).
This generalizes Proposition 3 of MVM2 and says that 1) the probability at the best design reliability state will decrease after a failed test but will increase after a successful test, and 2) the probability at the worst design reliability state will increase after a failed test but will decrease after a successful test.

3.2 Properties of the Update of $s$ After a Redesign

Simple properties of stationary finite state Markov chains can be used to establish some properties of the effects of redesign generalizing the 2-state Propositions 4 and 5 of MVM2.

**Proposition 4** (Probability distribution over the states, $(s_1, s_2, ..., s_k)$, after a single redesign)

**Case 1** ($u_{ij} > 0$ for $i \leq j$ and $u_{ij} = 0$ for $i > j$): For any $i$, \( \sum_{j=1}^{i} \delta_j(s) \leq \sum_{j=1}^{i} s_j \).

**Case 2** ($u_{ii} = 1$ for all $i = 1, 2, ..., k$): \( \delta(s) = s \); Redesign has no effect on $s$.

We will call the situation of Case 1 that of "non-regressive redesigns" following MVM2. In Case 1, Proposition 4 says that the distributions $\delta(s)$ and $s$ are stochastically ordered (and so, for example, $\delta_k(s) \geq s_k$ and $\delta_1(s) \leq s_1$).

**Proposition 5** (Probability distribution over the states $(s_1, s_2, ..., s_k)$ under an infinite sequence of redesigns)

**Case 1** The Markov chain transition matrix $u$ is irreducible, positive recurrent, and aperiodic: Under an infinite sequence of redesigns $s$ converges to a steady-state probability vector $\bar{\beta} = (\beta_1, ..., \beta_k)$ that may be computed by solving the linear equations $\bar{\beta} = \bar{\beta} \cdot u$ and $\beta_1 + \beta_2 + ... + \beta_k = 1$.

**Case 2** ($u_{ij} > 0$ for $i \leq j$ and $u_{ij} = 0$ for $i > j$): Under an infinite sequence of redesigns $s_k$ converges to 1 and expected design reliability $r(s)$ converges to $r_k$.

The Case 2 result says that if one makes an infinite sequence of non-regressive redesigns, eventually design reliability will be at the best design reliability state.

3.3 Properties of $V_n(s)$

The following two results are direct generalizations of the 2-state Propositions 10 and 11 of MVM2 (and have exactly the same proofs).

**Proposition 6** $V_n$ is monotone nondecreasing in $n$.

**Proposition 7** $V_n(s)$ is piecewise linear and convex in $s$.

4. ANALYSIS OF OPTIMAL DEVELOPMENT PLANS

The properties of our model recorded in Section 3 are important, but don’t directly help us identify or study the behavior of optimal development plans. In this section we first state those properties of optimal development that allow their computation and then describe how we used those properties and simulations in our analysis of the plans.
4.1 Optimal Next Actions and Evaluating $V_n(s)$

**Proposition 8** If $n < 1 + d$, stopping is an optimal next action and $V_n(s) = [n] \cdot r(s)$.

This proposition says that redesign or testing will not be beneficial if the remaining budget after making a redesign ($n - d$) would be below that required to build one system. This guarantees that eventually a development program will terminate and at least one system of the final design will be built.

**Proposition 9** If $n < 1 + t + d$, only stopping and redesign are potentially optimal next actions and

$$V_n(s) = \max_{l \geq 0} \{ [n - l \cdot d] \cdot r(\delta_l(s)) \}.$$ 

This proposition says that testing is not beneficial if the remaining budget after making a test would not be sufficient to purchase at least one redesign and build one system. It is better to stop or do a number of redesigns that produces the maximum expected payoff.

The following is simply a formalization of display (2.6) and says that for large current budgets, stopping, testing and redesign are all potential next actions.

**Proposition 10** If $n \geq 1 + t + d$, stopping, testing, and redesign are potentially optimal next actions and

$$V_n(s) = \max \{ [n] \cdot r(s), q(s) \cdot V_{n-t}(\eta_0(s)) + (1 - q(s)) \cdot V_{n-d}(\eta_1(s)), V_{n-d}(\tilde{\delta}(s)) \}.$$ 

4.2 Analysis of Optimal Development Programs

Our analysis of optimal development programs consists of 2 steps. First, using the results of Section 4.1 we compute and store $V_m(s)$ at each possible remaining budget point $m$, over a grid of probability distributions for the states (values of $s$). Second, we investigate the behavior of the optimal plans using simulation. During the simulation process, an optimal activity at any current budget $c_n$ and current probability vector for the states $s_n$ is determined using the information stored in the first step.

4.2.1 Computation of the Optimal Plans and Expected Payoff for an Initial Budget of $n$

The computation of optimal returns $V_m(s)$ at each possible remaining budget point ($m$) for all possible probability distributions over the states ($s$) proceeds by “backwards induction.” This process moves from the smallest to the largest possible remaining budget point. Inputs are the redesign transition matrix $u$, the initial budget $n$, the test cost $t$, the redesign cost $d$, the design reliability vector $r$, and parameters for a $(k, a)$-simplex-lattice design specifying the grid of vectors $s$ over which optimal payoffs will be evaluated.

In our first step we:

a) determine all possible remaining budget points, $m$, that might be reached in the development process using

$$m = n - k_1 \cdot t - k_2 \cdot d \geq 1,$$

where $k_1$ and $k_2$ (respectively a number of tests and a number of redesign in the sequence) are nonnegative integers,

b) sort the possible remaining budget points in ascending order

$$1 \leq m_1 < m_2 < \ldots < m_b = n,$$
where \( b \) is the total number of possible remaining budget points,

c) recursively determine optimal returns \( V_m(s) \) for all \( s \) on a grid by applying Propositions 8-10 (and interpolations where needed), starting from \( m_1 \) and proceeding to \( m_b = n \). In this process:

1) The grids of probability distributions \( (s) \) over the states are generated as elements of a \((k, a)\)-simplex lattice design. So each component of \( s \) is a multiple of \( a^{-1} \), and \( s_1 + s_2 + \ldots + s_k = 1 \). We used \( a = 5,000 \) in our analysis.

2) The procedure for determining optimal returns \( V_m(s) \) is:

\[
\begin{align*}
\text{If } 1 = m < 1 + t, & \quad V_m(s) = \Psi_1, \\
\text{if } 1 + d = m < 1 + t + d, & \quad V_m(s) = \max \{ \Psi_1, \Psi_2 \}, \\
\text{if } 1 + t + d = m, & \quad V_m(s) = \max \{ \Psi_1, \Psi_2, \Psi_3 \},
\end{align*}
\]

where

\[
\begin{align*}
\Psi_1 &= |m| \cdot r(s), \\
\Psi_2 &= q(s) \cdot \text{INTERP} \left[ V_{m-d}(\bar{m}_0(s)) \right] + (1 - q(s)) \cdot \text{INTERP} \left[ V_{m-d}(\bar{m}_1(s)) \right], \\
\Psi_3 &= \text{INTERP} \left[ V_{m-d}(\bar{d}(s)) \right],
\end{align*}
\]

and the interpolation method \( \text{INTERP}[\] is described in Appendix A.5.

4.2.2 Simulating the Behavior of an Optimal Development Plan

We study the behavior of development plans using simulation. Simulation of the development plan for an initial probability distribution, \( S_0 \), and an initial budget of \( n \) involves randomly generating test results and the effects of redesigns. During the simulation process, an optimal next activity at any point is determined by consulting the optimal returns stored as described in Section 4.2.1.

One simulation “trial” runs as follows. Inputs are an initial probability distribution over the states \( S_0 \) and the optimal returns contained in Table of \( V_m(s) \) generated as in Section 4.2.1. Then:

a) An initial reliability state is generated according to the initial distribution over the states \( S_0 \).

b) Starting at initial budget \( n \) and initial probability vector \( S_0 \), an optimal next activity at any current budget \( n_c \) and current probability vector for states \( S_c \) is determined by using Propositions 8-10 and the stored values of \( V_m(s) \).

c) Depending upon what activity is prescribed in (b), the current budget \( n_c \), the probability vector \( S_c \), and the current reliability state are updated as follows:

1) If the activity prescribed is “redesign,” the budget is reduced to \( n' = n_c - d \), the probability vector is updated to \( s' = \delta(s_c) \) and a new real reliability state is generated from the current one using the transition matrix \( u \).

2) If the activity prescribed is “test,” the budget is reduced to \( n' = n_c - t \), the probability vector is updated to \( s' = \pi_0(S_c) \) or \( s' = \pi_1(S_c) \) depending on a test result which is generated by using the current real design reliability \( (\Pi) \) and the real design reliability is not changed.

d) The development process is terminated when “build” becomes an optimal next activity, and we record the conditional expected number of effective systems built \( B^* \cdot \Pi^* \), for \( B^* \) the final remaining budget and \( \Pi^* \) the final realized design reliability.
Steps a) through d) are repeated until a desired number of trials are reached.

5. SOME NUMERICAL RESULTS

5.1 \(k\)-State Results Without Accelerated Testing (the Effects of Model Parameters on Optimal Plan Behavior)

In this section we consider how the factors, test cost \((t)\), redesign cost \((d)\), the redesign transition matrix \((u)\), and the design reliability vector \((r)\) affect the behavior and performance of optimal plans. Most of the discussion is based primarily on extensive simulation results (for a total of 7,128 different problems) using 3-state reliability models with \(q = r\) reported in Shevasuthisilp [6]. We have also done some simulations for 4- and 5-state models to verify that our methods and analyses are capable of handling larger numbers of states and produce qualitatively same results as for 3-state models.

Some representative results from the large set reported in Shevasuthisilp [6] are summarized in tables in this section. Model parameters and plan characteristics recorded here are: initial probability distribution for the states \((\bar{s}_0)\) and average probability distribution at program end \((\bar{s}^\ast)\), initial expected reliability \((\bar{r}(\bar{s}_0))\) and average expected reliability at program end \((\bar{r}(\bar{s}^\ast))\), the expected number of effective systems without \((V_0 = n \cdot \bar{r}(\bar{s}_0))\) and with \((\bar{V}^\ast)\) development, the optimal first action \((F, \text{where } 1 \text{ is "build", } 2 \text{ is "test", and } 3 \text{ is "redesign"})\) and the average numbers of systems built \((\bar{B}^\ast)\), redesigns made \((\bar{D}^\ast)\) and tests made \((\bar{T}^\ast)\). The averages in the tables come from 25,000 simulation trials per case. The particular forms of \(u\) referred to in the tables and their captions are given and discussed briefly in Appendix A.6.

Table 5.1 shows how the redesign cost affects the behavior of optimal plans. The test cost is fixed at 5 and redesign costs are \(d = 5\) and 50. We find that as the redesign cost increases, the average number of redesigns made by optimal plans decreases. This decreases the optimal mean number of effective systems built and the average final expected reliability. The findings are sensible, because when the cost of redesign is high, it is not economical to do many redesigns. Increasing the redesign cost also affects the number of tests made and the optimal first activity in a development program. As the redesign cost increases, the optimal first activity can change from test to build. It is not beneficial to do testing alone without following poor test results with redesign.

Table 5.1: Simulation Results for \(n = 1,000\), \(t = 5\), \(u(a, g = 0.05, f = 0.25)\), and \(q = r = (0.10, 0.50, 0.90)\)

<table>
<thead>
<tr>
<th>((s_{01}, s_{02}, s_{03}))</th>
<th>(d)</th>
<th>((\bar{s}_1, \bar{s}_2, \bar{s}_3))</th>
<th>(r(\bar{s}_0))</th>
<th>(\bar{r}(\bar{s}^\ast))</th>
<th>(V_0)</th>
<th>(\bar{V}^\ast)</th>
<th>(F)</th>
<th>(\bar{B}^\ast)</th>
<th>(\bar{D}^\ast)</th>
<th>(\bar{T}^\ast)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((1.00, 0.00, 0.00))</td>
<td>5</td>
<td>((0.000, 0.055, 0.945))</td>
<td>0.100</td>
<td>0.878</td>
<td>770.15</td>
<td>3</td>
<td>876.71</td>
<td>8.62</td>
<td>16.02</td>
<td></td>
</tr>
<tr>
<td>((0.072, 0.230, 0.698))</td>
<td>50</td>
<td>((0.072, 0.230, 0.698))</td>
<td>0.751</td>
<td>100.00</td>
<td>515.26</td>
<td>3</td>
<td>652.19</td>
<td>5.93</td>
<td>10.22</td>
<td></td>
</tr>
<tr>
<td>((0.000, 0.048, 0.952))</td>
<td>5</td>
<td>((0.000, 0.048, 0.952))</td>
<td>0.881</td>
<td>829.45</td>
<td>2</td>
<td>941.30</td>
<td>2.26</td>
<td>9.48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((0.000, 0.300, 0.700))</td>
<td>50</td>
<td>((0.000, 0.300, 0.700))</td>
<td>0.780</td>
<td>780.00</td>
<td>1</td>
<td>1000.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((0.000, 0.055, 0.945))</td>
<td>5</td>
<td>((0.000, 0.055, 0.945))</td>
<td>0.878</td>
<td>788.78</td>
<td>2</td>
<td>898.14</td>
<td>5.74</td>
<td>14.64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((0.034, 0.309, 0.657))</td>
<td>50</td>
<td>((0.034, 0.309, 0.657))</td>
<td>0.620</td>
<td>620.00</td>
<td>2</td>
<td>827.15</td>
<td>2.64</td>
<td>8.18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2 shows how the test cost \((t)\) affects the behavior of optimal plans. The redesign cost is fixed at \(d = 5\) and test costs are \(t = 5\) and 50. We find that as the test cost increases, the average number of tests made by optimal plans decreases. We also find that increasing the test cost reduces the optimal mean number of effective systems and the average final expected reliability. The findings are reasonable.
optimal plan uses few tests when testing is expensive. In such cases there is little empirical reliability information available for design improvement. It is also evident from Table 5.2 that the optimal first activity in a development program tends to change from test to build as test cost increases.

Table 5.2: Simulation Results for \( n = 1000, \ d = 5, \ u_a (g = 0.05, f = 0.25), \) and \( q = r = (0.10, 0.50, 0.90) \)

<table>
<thead>
<tr>
<th>((s_{01}, s_{02}, s_{03}))</th>
<th>(t)</th>
<th>((\bar{s}, \bar{s}, \bar{s}))</th>
<th>(r(s_0))</th>
<th>(\bar{r}(s))</th>
<th>(V_0)</th>
<th>(\bar{V})</th>
<th>(F)</th>
<th>(\bar{B})</th>
<th>(\bar{D})</th>
<th>(\bar{T})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((1.00, 0.00, 0.00))</td>
<td>5</td>
<td>(0.000, 0.555, 0.945)</td>
<td>0.878</td>
<td>100.00</td>
<td>770.15</td>
<td>3</td>
<td>876.71</td>
<td>8.62</td>
<td>16.02</td>
<td></td>
</tr>
<tr>
<td>((1.00, 0.00, 0.00))</td>
<td>50</td>
<td>(0.163, 0.229, 0.608)</td>
<td>0.678</td>
<td>482.09</td>
<td>829.45</td>
<td>2</td>
<td>941.30</td>
<td>2.26</td>
<td>9.48</td>
<td></td>
</tr>
<tr>
<td>((0.00, 0.30, 0.70))</td>
<td>5</td>
<td>(0.000, 0.048, 0.952)</td>
<td>0.881</td>
<td>780.00</td>
<td>780.00</td>
<td>1</td>
<td>1000.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>((0.00, 0.70, 0.30))</td>
<td>5</td>
<td>(0.000, 0.700, 0.300)</td>
<td>0.878</td>
<td>788.78</td>
<td>898.14</td>
<td>2</td>
<td>989.14</td>
<td>5.74</td>
<td>14.64</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3 shows how the redesign transition matrix affects the behavior of optimal plans. Redesign and test costs are fixed at \( t = d = 5 \). We find that when a more effective redesign transition matrix is used \( (u_b \) in place of \( u_a \)), the average number of redesigns made and amount of resources devoted to development \( (n - B^*) \) by optimal plans decrease, but the mean number of effective systems and final expected reliability increase. Using a better redesign transition matrix also decreases the number of tests made and tends to change the optimal first activity from test to redesign.

Table 5.3: Simulation Results for \( n = 1000, \) with \( q = r = (0.10, 0.50, 0.90) \), \( u_a (g = 0.05, f = 0.25), \) and \( u_b (g = 0.05, f = 0.75) \)

<table>
<thead>
<tr>
<th>((s_{01}, s_{02}, s_{03}))</th>
<th>(u)</th>
<th>((\bar{s}, \bar{s}, \bar{s}))</th>
<th>(r(s_0))</th>
<th>(\bar{r}(s))</th>
<th>(V_0)</th>
<th>(\bar{V})</th>
<th>(F)</th>
<th>(\bar{B})</th>
<th>(\bar{D})</th>
<th>(\bar{T})</th>
</tr>
</thead>
<tbody>
<tr>
<td>((1.00, 0.00, 0.00))</td>
<td>(u_a)</td>
<td>(0.000, 0.055, 0.945)</td>
<td>0.878</td>
<td>100.00</td>
<td>770.15</td>
<td>3</td>
<td>876.71</td>
<td>8.62</td>
<td>16.02</td>
<td></td>
</tr>
<tr>
<td>((1.00, 0.00, 0.00))</td>
<td>(u_b)</td>
<td>(0.000, 0.039, 0.960)</td>
<td>0.884</td>
<td>848.01</td>
<td>959.32</td>
<td>3</td>
<td>959.32</td>
<td>3.56</td>
<td>4.58</td>
<td></td>
</tr>
<tr>
<td>((0.00, 0.30, 0.70))</td>
<td>(u_a)</td>
<td>(0.000, 0.048, 0.952)</td>
<td>0.881</td>
<td>829.45</td>
<td>941.30</td>
<td>2</td>
<td>941.30</td>
<td>2.26</td>
<td>9.48</td>
<td></td>
</tr>
<tr>
<td>((0.00, 0.30, 0.70))</td>
<td>(u_b)</td>
<td>(0.000, 0.028, 0.972)</td>
<td>0.889</td>
<td>859.23</td>
<td>966.96</td>
<td>3</td>
<td>966.96</td>
<td>2.09</td>
<td>4.51</td>
<td></td>
</tr>
<tr>
<td>((0.00, 0.70, 0.30))</td>
<td>(u_a)</td>
<td>(0.000, 0.055, 0.945)</td>
<td>0.878</td>
<td>788.78</td>
<td>989.14</td>
<td>2</td>
<td>989.14</td>
<td>5.74</td>
<td>14.64</td>
<td></td>
</tr>
<tr>
<td>((0.00, 0.70, 0.30))</td>
<td>(u_b)</td>
<td>(0.002, 0.032, 0.966)</td>
<td>0.886</td>
<td>860.13</td>
<td>971.27</td>
<td>3</td>
<td>971.27</td>
<td>2.14</td>
<td>3.61</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4 summarizes simulation results for high reliability using redesign transition matrix \( u_b \), \( t = 5 \), and \( d = 5 \) and 50. The behavior of optimal plans is “unusual” for these cases, since no test is made in any optimal program. This behavior is consistent across all such high design reliability cases we studied (495 combinations of parameter using \( t = d = 5 \), 10 and 50, 4 redesign transition matrices, and (66) initial probability distributions). Since design reliability is always high (always at least 0.8), the likelihood of test failure in any reliability state is very low and testing does not produce much useful
information. Testing only wastes limited development resources if it does not provide a basis to discriminate effectively among reliability states.

Table 5.4: Simulation Results for the Set of Parameters: \( n = 1,000, \ t = 5, \ u_b (g = 0.05, f = 0.75), \) and \( g = r = (0.80, 0.85, 0.90) \)

<table>
<thead>
<tr>
<th>((s_{01}, s_{02}, s_{03}))</th>
<th>(d)</th>
<th>((\bar{s}_1, \bar{s}_2, \bar{s}_3))</th>
<th>(r(\bar{s}_0))</th>
<th>(\bar{V}^*)</th>
<th>(V_0)</th>
<th>(F)</th>
<th>(\bar{B}^*)</th>
<th>(\bar{D}^*)</th>
<th>(\bar{T}^*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (1.00,0.00,0.00) )</td>
<td>5</td>
<td>((0.210,0.163,0.628))</td>
<td>0.871</td>
<td>862.20</td>
<td>3</td>
<td>990.00</td>
<td>2.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>((0.287,0.356,0.356))</td>
<td>0.853</td>
<td>810.76</td>
<td>3</td>
<td>950.00</td>
<td>1.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>( (0.00,0.30,0.70) )</td>
<td>5</td>
<td>((0.000,0.300,0.700))</td>
<td>0.885</td>
<td>885.00</td>
<td>1</td>
<td>1000.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>((0.000,0.300,0.700))</td>
<td>0.885</td>
<td>885.00</td>
<td>1</td>
<td>1000.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>( (0.00,0.70,0.30) )</td>
<td>5</td>
<td>((0.000,0.700,0.700))</td>
<td>0.876</td>
<td>871.90</td>
<td>3</td>
<td>995.00</td>
<td>1.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>((0.000,0.700,0.300))</td>
<td>0.865</td>
<td>865.00</td>
<td>1</td>
<td>1000.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5 enables comparisons of the behavior of optimal plans for different design reliability vectors \((q = r = (0.10,0.30,0.50), (0.10,0.50,0.90), \) and \((0.80,0.85,0.90))\). We find that optimal programs for low design reliability problems employ more tests and redesigns than optimal programs for high design reliability problems. This finding agrees with intuition that when reliability is low, more development resources should be devoted to improve current design reliability.

Table 5.5: Simulation Results for \( n = 1,000, \ t = 5, \ d = 5, \) and \( u_a (g = 0.05, f = 0.25) \)

<table>
<thead>
<tr>
<th>((s_{01}, s_{02}, s_{03}))</th>
<th>(r)</th>
<th>(r(\bar{s}_0))</th>
<th>(\bar{V}^*)</th>
<th>(V_0)</th>
<th>(F)</th>
<th>(\bar{B}^*)</th>
<th>(\bar{D}^*)</th>
<th>(\bar{T}^*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (1.00,0.00,0.00) )</td>
<td>((0.10,0.30,0.50))</td>
<td>0.100</td>
<td>0.466</td>
<td>100.00</td>
<td>366.73</td>
<td>3</td>
<td>818.20</td>
<td>11.70</td>
</tr>
<tr>
<td></td>
<td>((0.80,0.85,0.90))</td>
<td>0.800</td>
<td>0.822</td>
<td>800.00</td>
<td>814.00</td>
<td>3</td>
<td>990.00</td>
<td>2.00</td>
</tr>
<tr>
<td>( (0.00,0.30,0.70) )</td>
<td>((0.10,0.30,0.50))</td>
<td>0.440</td>
<td>0.440</td>
<td>440.00</td>
<td>440.00</td>
<td>1</td>
<td>1000.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>((0.80,0.85,0.90))</td>
<td>0.885</td>
<td>0.885</td>
<td>885.00</td>
<td>885.00</td>
<td>1</td>
<td>1000.00</td>
<td>0.00</td>
</tr>
<tr>
<td>( (0.00,0.70,0.30) )</td>
<td>((0.10,0.30,0.50))</td>
<td>0.360</td>
<td>0.440</td>
<td>360.00</td>
<td>378.85</td>
<td>2</td>
<td>859.59</td>
<td>7.84</td>
</tr>
<tr>
<td></td>
<td>((0.80,0.85,0.90))</td>
<td>0.865</td>
<td>0.865</td>
<td>865.00</td>
<td>865.00</td>
<td>1</td>
<td>1000.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

In all, careful examination of our simulation results confirms that our mathematics is behaving qualitatively “exactly as it should,” in complete accord with intuition. What it provides is, of course, exact quantitative guidelines consistent with any set of input model parameters.

5.2 Relationship Between Computing Time and Number of Design Reliability States

Table 5.6 shows some average computing times for 3-state, 4-state, and 5-state models. The computing time has 2 parts. First is the set-up time needed to build a table containing optimal returns for all possible probability distributions at all possible remaining budget points. The set-up time for a given \( k \) and \( n \) is approximately constant in the other problem parameters and mostly depends on the number of \( s \) grid points used. Second, there is an average simulation time used to study a development plan. The average (across initial distributions \((\xi_0)\)) simulation times for all 25,000 trials of the development programs for 3,
4, and 5-state models are displayed (for 66, 56, and 70 initial distributions $S_0$ respectively). As we expect, the computing time increases roughly exponentially in the number of design reliability states.

Table 5.6: Computing Time for $t = d = 5$, $(r_1, r_2) = (0.10, 0.90)$, and $n = 1,000$

<table>
<thead>
<tr>
<th>Computing Time (Minutes)*</th>
<th>3-State Model</th>
<th>4-State Model</th>
<th>5-State Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set-up Time</td>
<td>39.000</td>
<td>314.000</td>
<td>2,058.000</td>
</tr>
<tr>
<td>Average Simulation Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for Problem 1 (using $u_a$)</td>
<td>0.197</td>
<td>1.697</td>
<td>9.685</td>
</tr>
<tr>
<td>for Problem 2 (using $u_b$)</td>
<td>0.061</td>
<td>0.411</td>
<td>2.871</td>
</tr>
</tbody>
</table>

*(Using an 800 MHz Pentium II computer with 512 MB RAM, running programs developed in C++)

5.3 $k$-State Results with Accelerated Testing

In general, the acceleration of testing described in Section 2.2 does affect the behavior of optimal development programs in high reliability problems. The effects are positive, since the accelerated testing provides more useful reliability information for guiding design improvement. In our high reliability cases, expected returns under accelerated testing were always higher than expected returns under the normal use testing conditions. We also found that in the cases we studied, the higher the acceleration factor, the stronger the positive effects. The optimal plans under acceleration are intuitively more practical and reasonable than without acceleration. Optimal plans change from employing only redesign, repeated redesign or immediate build, to using a mixed sequence of tests and redesigns.

Tables 5.7 and 5.8 (for $r = (0.80, 0.85, 0.90)$) and Table 5.9 (for $r = (0.900, 0.945, 0.990)$) summarize numbers of cases (at 66 initial probability distributions $S_0$ with grid size of 0.10) affected by acceleration of testing. If optimal plans are affected (changed by the use of $a_f > 1$), the differences $(I)$ between expected optimal returns under accelerated testing and normal use conditions testing are summarized. (As in our other simulations, 25,000 runs were made for each case.)

Table 5.7 shows how the behavior of optimal plans is affected by using the maximum possible acceleration factor at different levels of redesign costs. The number of optimal plans affected decreases as the redesign cost $(d)$ increases. This implies that accelerated testing is less effective as the redesign cost increases. Even more informative testing alone is not beneficial if redesign cannot be economically made after testing. When the redesign cost is high, one is pushed toward an initial “build” action.

Table 5.8 shows how the behavior of optimal plans is affected by using the maximum possible acceleration factor for three different redesign transition matrices. Most of the optimal plans are affected when redesign transition matrix is $u_b (g = 0.05, f = 0.75)$, a redesign transition matrix describing moderately effective redesigns. Accelerated testing has almost no effect when redesigns are highly effective (for the matrix $u_a (g = 0.05, f = 1.00)$ redesign always improves system reliability). When redesign is highly effective, redesign always dominates the effect of (even accelerated) testing. Accelerated testing has less effect on the optimal plans for $u_a (g = 0.05, f = 0.25)$ than for $u_b (g = 0.05, f = 0.75)$, because the redesign transition matrix $u_a$ describes the least effective redesign.
mechanism. For this case redesign is rarely an optimal activity and (accelerated) testing is not called for either, since testing alone is not beneficial if it is not followed by effective redesign.

Table 5.7: Results for 66 Distributions $x_0$ and Parameters $n = 1000$,

\[ t = 5, u_s (g = 0.05, f' = 0.75), r = (0.80,0.85,0.90), \text{ and } a_f = 5 \text{ (the maximum possible)} \]

<table>
<thead>
<tr>
<th>Numbers of Cases</th>
<th>$d = 5$</th>
<th>$d = 10$</th>
<th>$d = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unchanged Optimal Plans</td>
<td>10 (15.15%)</td>
<td>27 (40.91%)</td>
<td>66 (100%)</td>
</tr>
<tr>
<td>Changed Optimal Plans</td>
<td>56 (84.85%)</td>
<td>39 (59.09%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>with $0 &lt; I \leq 5$</td>
<td>51</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>With $5 &lt; I \leq 10$</td>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

($I$ is the increase in the mean final number of effective systems produced by acceleration.)

Table 5.8: Results for 66 Distributions $x_0$ and Parameters $n = 1000, g = 0.05$

\[ t = 2.5, d = 5, r = (0.80,0.85,0.90), \text{ and } a_f = 5 \text{ (the maximum possible)} \]

<table>
<thead>
<tr>
<th>Numbers of Cases</th>
<th>$u_s (f = 0.25)$</th>
<th>$u_s (f = 0.75)$</th>
<th>$u_s (f = 1.00)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unchanged Optimal Plans</td>
<td>41 (62.12%)</td>
<td>4 (6.06%)</td>
<td>62 (93.94%)</td>
</tr>
<tr>
<td>Changed Optimal Plans</td>
<td>25 (37.88%)</td>
<td>62 (93.94%)</td>
<td>4 (6.06%)</td>
</tr>
<tr>
<td>with $0 &lt; I \leq 5$</td>
<td>8</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>with $5 &lt; I \leq 10$</td>
<td>7</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>with $10 &lt; I \leq 15$</td>
<td>10</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>with $15 &lt; I \leq 20$</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

($I$ is the increase in the mean final number of effective systems produced by acceleration.)

Table 5.9 shows how the behavior of optimal plans is affected the acceleration factor ($a_f$). The number of optimal plans affected and performance measures increase as the acceleration factor increases. This is intuitively appealing because testing under high acceleration provides more informative design reliability information.
Table 5.9: Results for 66 Distributions $s_0$ and Parameters $n = 1000$, $t = 2.5$, $d = 5$, and $r = (0.900, 0.945, 0.990)$

<table>
<thead>
<tr>
<th></th>
<th>Number of Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a_f = 7$</td>
</tr>
<tr>
<td>Unchanged Optimal Plans</td>
<td>5 (7.57%)</td>
</tr>
<tr>
<td>Changed Optimal Plans</td>
<td>61 (92.43%)</td>
</tr>
<tr>
<td>with $0 &lt; I \leq 5$</td>
<td>7</td>
</tr>
<tr>
<td>with $5 &lt; I \leq 10$</td>
<td>51</td>
</tr>
<tr>
<td>with $10 &lt; I \leq 15$</td>
<td>3</td>
</tr>
<tr>
<td>with $15 &lt; I \leq 20$</td>
<td>0</td>
</tr>
</tbody>
</table>

($J$ is the increase in the mean final number of effective systems produced by acceleration.)

6. CONCLUSIONS AND FINAL COMMENTS

The purpose of our study has been to identify optimal development programs for one-shot systems with the goal of attaining high final design reliability while spending as little of a fixed budget as possible. Our model is an extension of the 2-state model of Moon, Vardeman, and McBeth [1999]. We generalized their theories and analyses to cover any finite number of design reliability states. A model using a larger number of design reliability states provides the possibility of more refined modeling, but computing time increases exponentially with the number of states. Using a larger number of reliability states also requires more initial inputs (the more detailed initial probability distribution for the set of design reliabilities and especially a defensible form for $w$), which may be difficult and costly to determine objectively. As is always the case when contemplating the use of nested mathematical models, a balance must be struck between increasing potential model fidelity given appropriate values for increasing numbers of parameters and one’s diminishing ability to adequately specify them.

We investigated how the 1) test cost, 2) redesign cost, and 3) redesign transition matrix affect the behavior of 3-state optimal plans and found that those factors have intuitively plausible effects. We find that “optimal development programs providing large expected numbers of effective systems can be obtained, if testing is not expensive and can provide informative results for discriminating among reliability states and redesign uses the information to correct system faults effectively.”

We also investigated how the design reliability vector $r$ affects the behavior of optimal plans. We conclude that 1) optimal programs for low design reliability employ more development resources (more tests and redesigns) than optimal programs for high design reliability and 2) the behavior of optimal programs for ultra high design reliability is “unusual” in that optimal plans for all such cases considered involved no testing. Since design reliability is very high for all states, testing under normal use conditions is not beneficial because it does not produce useful information for discriminating between states.

The intuitively unappealing behavior of optimal development programs for high design reliability problems inspired us to allow $q \neq r$. We considered the possibility of accelerated testing, where test failure probabilities are a fixed multiple of design failure probabilities under conditions of normal use. Simulation results for the model allowing accelerated testing are promising. Testing is part of some optimal...
plans. The accelerated testing has “positive” consequences such as producing more appealing optimal plans, improving performance measures, and increasing the number of tests and redesigns made by optimal plans. We also find that the larger the acceleration factor used, the stronger the positive effects on optimal plans. (The maximum possible acceleration factor is \((1 - r_i)^{-1}\).) But accelerated testing alone is not beneficial if redesign is not effective. “Therefore to obtain a large optimal return, we need both informative testing providing useful information on current design reliability and effective redesign that uses this information to correct system faults.”

After determining appropriate acceleration factors that theoretically provide high optimal returns and reasonable development plan behavior, users must determine how to link the desire acceleration constants to a real physical test strategy. Physical mechanisms like increased temperature, voltage, or pressure that increase failure probabilities must be identified and their effects accurately quantified. We also note that a different relationship between \(q\) and \(r\) than our \(q_i = 1 - a_f (1 - r_i)\) from Section 2.2 could easily be appropriate and call for revisiting the “accelerated testing” computations.

There are further issues that can be addressed to improve our current analyses. Among these are the following.

a) We have used a simple stationary Markov chain transition mechanism to describe the effects of redesign. In reality designers could gain experience over time or it might be very difficult to redesign effectively late in a development program. The effects of redesign might thus be better described using dynamic redesign transition matrices. They might change over time, with the number of redesigns made, the average design reliability, or according to a specific type of design flaw identified in testing.

b) The cost of redesign in the current analysis is constant. But there are different types of design problems, which need different corrective actions. Therefore cost of redesign might in reality not be constant. It is then potentially more realistic to describe it as a function of time, the type of redesign transition matrix applied, or the current distribution over design reliability states.

c) Testing used in our work provides only Bernoulli results: pass or fail. These may fail to be informative enough. Other probability distributions might be used to describe test results. Poisson, Normal, or Gamma distributions might be employed (with “test failure” defined in terms of such a variable but the measured value of the variable available from the test).

d) Testing under accelerated conditions may consume more resources than testing under normal use conditions and accelerated testing costs should perhaps increase with the acceleration factor. In our present analyses, we used a test cost that was constant in \(a_f\). It might be more realistic to consider models where the test cost is a function of the acceleration factor.

A. Appendix

In this appendix we provide proofs for several of the propositions. (We present only those proofs that are both fundamentally different from any presented in MVM2 for 2-state cases and also perhaps not completely obvious.) We then give some details for the interpolation method we used in our computations and specify the redesign transition matrices we employed in our numerical work.

A.1 Proof of Proposition 3
Consider showing that \(n_{1k}(s) \leq s_k\).

\[
s_k - n_{1k}(s) = s_k - \frac{s_k \cdot (1 - r_k)}{1 - (r_1 \cdot s_1 + r_2 \cdot s_2 + ... + r_k \cdot s_k)}
\]
\[
\begin{align*}
    &\quad = s_k \cdot \left[ r_k \cdot \left( 1 - s_k \right) - r_1 \cdot s_1 - r_2 \cdot s_2 - \ldots - r_{k-1} \cdot s_{k-1} \right] \\
    &\quad = s_k \cdot \left[ r_k \cdot \left( s_1 + s_2 + \ldots + s_{k-1} \right) - r_1 \cdot s_1 - r_2 \cdot s_2 - \ldots - r_{k-1} \cdot s_{k-1} \right] \\
    &\quad = s_k \cdot \left[ \left( s_1 \cdot \left( r_k - r_1 \right) + s_2 \cdot \left( r_k - r_2 \right) + \ldots + s_{k-1} \cdot \left( r_k - r_{k-1} \right) \right) \right] \\
    &\quad = \frac{s_k \cdot \left[ \left( s_1 \cdot \left( r_k - r_1 \right) + s_2 \cdot \left( r_k - r_2 \right) + \ldots + s_{k-1} \cdot \left( r_k - r_{k-1} \right) \right) \right]}{1 - r(s)}.
\end{align*}
\]

(A.1)

Since \( r_k > r_{k-1} > \ldots > r_1 \), the expression on the right of (A.1) is greater than or equal to 0.

The arguments for the inequalities \( \eta_{00}(s) \geq s_k, \eta_{01}(s) \leq s_1 \) and \( s_1 \leq \eta_{11}(s) \) are analogous. \( Q.E.D. \)

A.2 Proof of Proposition 4 (Case I)

For the non-regressive case

\[
\sum_{j=1}^{i} \delta_j(s) = \sum_{j=1}^{i} \left( \sum_{l=1}^{j} s_l u_l \right) = \sum_{j=1}^{i} \sum_{l=1}^{j} s_l u_l = \sum_{j=1}^{i} s_l \sum_{j=1}^{l} u_j \leq \sum_{j=1}^{i} s_j .
\]

Q.E.D.

A. 3 Proof of Proposition 8

For \( t \geq d \): Here \( n < 1 + t \) so making either a test or redesign will reduce the current budget below that required to build at least one system of the final design. Therefore stopping is an optimal next action.

For \( t < d \):

Case \((n < 1 + d) \text{ AND } (n < 1 + t)\): Making either a redesign or a test will reduce the budget below that required to build at least one system of the final design. Therefore stopping is an optimal next action.

Case \((n < 1 + d) \text{ AND } (1 + t < n < 1 + 2t)\): Stopping and making a test are potentially optimal next actions.

\[
V_n(s) = \max \{ [n] \cdot r(s), q(s) \cdot V_{n-t}(\eta_{00}(s)) + (1 - q(s)) \cdot V_{n-t}(\eta_{11}(s)) \}
\]

\[
= \max \{ [n] \cdot r(s), q(s) \cdot [n - t] \cdot r(\eta_{00}(s)) + (1 - q(s)) \cdot [n - t] \cdot r(\eta_{11}(s)) \}
\]

Apply Proposition 1 and this becomes

\[
V_n(s) = \max \{ [n] \cdot r(s), [n - t] \cdot r(s) \},
\]

\[
\vdots
\]

Case \((n < 1 + d) \text{ AND } (1 + (k-1) \cdot t < n < 1 + k \cdot t), \text{ for a positive integer } k\): The expected payoffs can be determined by induction and applying Proposition 1. Therefore we can conclude that stopping is an optimal next action. \( Q.E.D. \)

A. 4 Proof of Proposition 9

For \( t \leq d \): The potential optimal next options are stopping, redesign, or testing and the optimal return is from (2.6)
\[ V_n(\vec{s}) = \max \{ n \cdot r(\vec{s}), q(\vec{s}) \cdot V_{n-t}(\vec{m}_0(\vec{s})), (1-q(\vec{s})) \cdot V_{n-t}(\vec{m}_1(\vec{s})), V_{n-d}(\vec{\delta}(\vec{s})) \}. \]

An optimal next action after making a test is stopping (using Proposition 8, since the remaining budget after making a test is less than \( 1+ d \)). Thus
\[
V_n(\vec{s}) = \max \{ n \cdot r(\vec{s}), q(\vec{s}) \cdot (n-t) \cdot r(\vec{m}_0(\vec{s})), (1-q(\vec{s})) \cdot (n-t) \cdot r(\vec{m}_1(\vec{s})), n-d \cdot r(\vec{\delta}(\vec{s})) \}
\]
\[ = \max \{ n \cdot r(\vec{s}), q(\vec{s}) \cdot r(\vec{m}_0(\vec{s})), (1-q(\vec{s})) \cdot r(\vec{m}_1(\vec{s})), n-d \cdot r(\vec{\delta}(\vec{s})) \}. \]

Apply Proposition 1 to the second term. The recursion (2.6) becomes
\[
V_n(\vec{s}) = \max \{ n \cdot r(\vec{s}), n-t \cdot r(\vec{s}), n-t \cdot r(\vec{\delta}(\vec{s})) \}.
\]
The first term is greater than the second term. Therefore, potentially optimal next actions are stopping and redesign.

For \( t > d \):

- **Case** \( n < 1+t+d \) AND \( 1+d \leq n < 1+2d \): By the same argument as used in the \( t \leq d \) case, testing is not an option. Thus, potential actions are stopping and redesign and the expected payoffs are \( n \cdot r(\vec{s}) \) and \( V_{n-d}(\vec{\delta}(\vec{s})) \) respectively. Then apply Proposition 6 to the second term and it becomes \( n-d \cdot r(\vec{\delta}(\vec{s})) \),
so \( V_n(\vec{s}) = \max_{t=0,1} \{ n-l \cdot d \cdot r(\vec{\delta}(\vec{s})) \}. \)

- **Case** \( n < 1+t+d \) AND \( 1+ (l-1) \cdot d \leq n < 1+l \cdot d \) for a positive integer \( l \): Again testing is not an option, and potential next actions are stopping and redesign and the expected payoffs are:
\[
V_n(\vec{s}) = \max \{ n \cdot r(\vec{s}), V_{n-d}(\vec{\delta}(\vec{s})) \}, \text{ and by induction}
\]
\[
V_{n-d}(\vec{\delta}(\vec{s})) = \max_{0 \leq m \leq l-1} \{ (n-d) - m \cdot d \cdot r(\vec{\delta}(\vec{s})) \}
\]

So
\[
V_{n-d}(\vec{\delta}(\vec{s})) = \max_{l \geq 0, s.t. n-l \cdot d \geq 1} \{ n - k \cdot d \cdot r(\vec{\delta}(\vec{s})) \} \text{ and}
\]
the possible options are stopping or doing at most \( \left\lfloor \frac{n-1}{d} \right\rfloor \) redesigns. \( \text{Q.E.D.} \)

### A.5 The Interpolation Method

Interpolation is needed during the process of recursively determining expected returns, \( V_n(\vec{s}) \), where at budget size of \( n \) and probability vector \( \vec{s} \) an updated probability distribution \( (\vec{s}') \) does not match exactly any point on the available grid of probability vectors.

We use multidimensional linear interpolation. We can hope that it will often be very accurate in our application, since Proposition 10 says that \( V_n(\vec{s}) \) is piecewise linear. The following is a complete description of our method for the \( k = 3 \) case. (Details for larger \( k \) are similar.)

Let \( V_n(\vec{s}) \) be a value to be interpolated. Write \( \vec{s} = (s_1, s_2, s_3) \) and think of \( V_n(\vec{s}) \) as a function of \( s_2 \) and \( s_3 \), and already evaluated for those \( \vec{s} \) where entries of \( \vec{s} \) are multiples of \( \frac{1}{a} \). Let
\[
pt_2 = s_2 \cdot a,
\]
and
\[
pt_3 = s_3 \cdot a.
\]
Define both $pt_2^-$ and $pt_3^+$ as $pt_2$ if $pt_2$ is an integer, and as the two consecutive integers with $pt_2^- \leq pt_2 \leq pt_2^+$ if $pt_2$ is not an integer. Similarly define $pt_3^-$ and $pt_3^+$.

- If $pt_2^+ + pt_3^+ \leq a$
  \[
  \text{Interp} \left[ V_n(s_2, s_3) \right] = (1 - f_a) \cdot (1 - f_v) \cdot V_n\left( \frac{pt_2^-}{a}, \frac{pt_3^-}{a} \right) + (f_a) \cdot (1 - f_v) \cdot V_n\left( \frac{pt_2^+}{a}, \frac{pt_3^-}{a} \right) \\
  + (1 - f_a) \cdot (f_v) \cdot V_n\left( \frac{pt_2^-}{a}, \frac{pt_3^+}{a} \right) + (f_a) \cdot (f_v) \cdot V_n\left( \frac{pt_2^+}{a}, \frac{pt_3^+}{a} \right),
  \]
  where
  \[
  f_a = 0, \text{ if } pt_2^- = pt_2^+ \text{ and otherwise } f_a = \frac{pt_2^+ - pt_2^-}{pt_2^+ - pt_2^-},
  \]
  and
  \[
  f_v = 0, \text{ if } pt_3^- = pt_3^+ \text{ and otherwise } f_v = \frac{pt_3^+ - pt_3^-}{pt_3^+ - pt_3^-}.
  \]
- If $pt_2^+ + pt_3^+ > a$,
  \[
  \text{Interp}[V_n(s_2, s_3)] = [1 - (pt_2 - pt_2^-) - (pt_3 - pt_3^-)] \cdot V_n\left( \frac{pt_2^-}{a}, \frac{pt_3^-}{a} \right) + \\
  (pt_2 - pt_2^-) \cdot V_n\left( \frac{pt_2^-}{a}, \frac{pt_3^+}{a} \right) + (pt_3 - pt_3^-) \cdot V_n\left( \frac{pt_2^+}{a}, \frac{pt_3^-}{a} \right).\]

\[A. \] **Transition Matrices Describing Effects of Redesigns**

In our simulations, the Markov chain transition matrices ($u$) describing the effects of redesigns are characterized by 2 parameters: 1) the diagonal probability ($g$), and 2) a conditional probability of improving design reliability given a reliability change ($f$). The diagonal probability represents a probability of staying at the same state through redesign. The fraction $f$ represents the fraction of changes that are improvements in design reliability.

Using the following forms for $u$, two "non-regressive" ($f = 1$) and two other transition matrices with possibility of design degradation ($f < 1$) were created by using $g = 0.05$ and $f = 0.25, 0.75$ and using $g = 0.05, 0.50$ and $f = 1.00$ respectively.

For $k = 3$

\[
\begin{pmatrix}
  g + (1 - f) \cdot (1 - g) & f \cdot (1 - g) / 2 & f \cdot (1 - g) / 2 \\
  (1 - f) \cdot (1 - g) / 2 & g & f \cdot (1 - g) / 2 \\
  (1 - f) \cdot (1 - g) / 2 & (1 - f) \cdot (1 - g) / 2 & g + f \cdot (1 - g)
\end{pmatrix}
\]
For $k = 4$

\[
\begin{bmatrix}
\frac{f \cdot (1-g)}{3} & \frac{f \cdot (1-g)}{3} & \frac{f \cdot (1-g)}{3} \\
\frac{g}{3} & \frac{f \cdot (1-g)}{2} & \frac{f \cdot (1-g)}{2} \\
\frac{(1-f) \cdot (1-g)}{3} & \frac{(1-f) \cdot (1-g)}{2} & \frac{(1-f) \cdot (1-g)}{2} \\
\frac{(1-f) \cdot (1-g)}{3} & \frac{(1-f) \cdot (1-g)}{3} & \frac{(1-f) \cdot (1-g)}{3} & \frac{g + f \cdot (1-g)}{3}
\end{bmatrix}
\]

For $k = 5$

\[
\begin{bmatrix}
\frac{f \cdot (1-g)}{4} & \frac{f \cdot (1-g)}{4} & \frac{f \cdot (1-g)}{4} & \frac{f \cdot (1-g)}{4} \\
\frac{(1-f) \cdot (1-g)}{4} & \frac{f \cdot (1-g)}{3} & \frac{f \cdot (1-g)}{3} & \frac{f \cdot (1-g)}{3} \\
\frac{(1-f) \cdot (1-g)}{3} & \frac{(1-f) \cdot (1-g)}{2} & \frac{(1-f) \cdot (1-g)}{2} & \frac{(1-f) \cdot (1-g)}{2} \\
\frac{(1-f) \cdot (1-g)}{4} & \frac{(1-f) \cdot (1-g)}{3} & \frac{(1-f) \cdot (1-g)}{3} & \frac{(1-f) \cdot (1-g)}{3} & \frac{g + f \cdot (1-g)}{4}
\end{bmatrix}
\]

REFERENCES