An Optimum, System-Based Component Testing Approach for Evaluating Software Reliability

Jayant Rajgopal, Mainak Mazumdar

Department of Industrial Engineering
University of Pittsburgh
Pittsburgh, PA 15261

Abstract: In a component testing approach for evaluating system reliability, one tests units of the components or subsystems that make up a larger system in order to draw conclusions about the reliability of the latter. An *optimum, system-based* component test plan explicitly account for the relationship between component and system reliabilities and also minimizes total testing effort. This paper reviews the main features of such plans and describes an application to reliability testing of a software system that is made up of several different components/modules.

1 Introduction

A dedicated program of testing is a crucial prerequisite for the design, development and eventual deployment of complex and highly reliable systems. Examples include mechanical, electrical/electronic, hardware and software systems. The objective of the test program is to demonstrate that the system will perform at a level of reliability that is acceptable with respect to the mission for which it is designed. These tests are required to demonstrate at some specified level of confidence that the system reliability exceeds some preassigned minimum. The reliability parameter for a generic system could be expressed as (1) a mean time to failure, (2) as the probability of no failures over a specified mission time such as a warranty period, or (3) the probability of no failure for a randomly chosen operating condition from the set of all conditions for which the system is designed. In particular, for software systems one might like the test plan to demonstrate that (a) the system will (with a high probability) provide failure-free operation for some specified mission time under specific conditions, or (b) that it will (with a high probability) process a randomly chosen application instance correctly.

Reliability tests for a system can be conducted either exclusively or in combination at various levels of assembly, such as component/module, subsystem, or system. They can also be conducted under different conditions, in different locations, or at different times. The most straightforward way to evaluate system reliability is to assemble the complete system and then test it as required. However, this is often undesirable (or even infeasible) because system-level tests can be prohibitively expensive due to the time and effort required to first assemble the system prior to testing, and because testing a system tends to be inherently more complex in terms of resources, facilities, instrumentation, etc., than testing component or simpler subsystems.

Before looking at the practical advantages of component level testing, some theoretical justification is also in order. We make the observation that in reliability theory there is a fairly extensive literature on the development of models to express system reliability as a function of component reliabilities (see for example, [BP75]). If one accepts the validity of such models, it follows logically that it should be possible to use results from component tests to make a valid inference on system reliability. This fact has not entirely been lost on other researchers, and some literature exists (e.g., [MSS74]) on obtaining interval estimates of system reliability based on component test data. However, the *design* aspects of the component test plans that yield these data have received relatively little attention, and this is the aspect that our research has sought to emphasize. The objective is to determine how total resources should be allocated either among tests of different components comprising a system or between components and the entire system, so that inferences on the system reliability can be made in the most cost-effective manner. This is an area where *ad hoc* decisions appear to be the norm and there is little evidence that statistics and optimization are used effectively in practice.

Using component tests either exclusively or in conjunction with system level tests in order to evaluate system reliability has numerous practical advantages:

- They generally result in lower test costs and a shorter overall test schedule.
- They provide more insight and information because more aspects of performance can be instrumented at a component level than would be possible at a system level.
- Component testing can be conducted by different organizational entities or development teams, and proceed at different locations and different times.
- It might be possible to postpone assembly of the entire system until its reliability can be guaranteed.

In addition to these general advantages, system-based component tests are particularly useful in the following situations:

Systems Mixing Old and New Components: Many systems are designed by combining new components and/or subsystems together with others which have been successfully used before in earlier designs. A good example of this would be software, where existing modules are often re-used between different software systems or different versions of the same system. Component/modular testing to demonstrate a system reliability requirement is particularly effective here: for a subsystem or module that has been used previously, little or no additional testing may be necessary, and only the new components might need more testing (of course, from an overall system perspective).

Continuously Evolving Designs: Many systems follow an evolutionary pattern where design improvement is continuous as technology advances. The product configuration is temporarily fixed at some discrete point in time and the configuration is released into the market. However, in actuality, the design process is continuous. Again, software presents a good example: earlier versions are often subsumed by succeeding ones that provide additional functionality. Component test plans are a good option in such cases. When a particular component changes to the extent that it renders the previous version obsolete, reliability tests can be conducted on it. Data from these tests can then be

combined with the most recent test results of other components in order to evaluate the reliability of the new system.

Complex Multi-Function Systems: With many complex systems, it is often the case that as part of the overall project diverse design teams will have been assembled in different locations and be working according to different (but coordinated) schedules. Many large-scale software systems follow this model. Component test plans represent a more practical approach to system reliability evaluation in these situations. It might be possible to run tests independently at different locations, and the data can then be pooled without having to first assemble the system into its final configuration.

The factors discussed above have been recognized in industry and component tests are carried out for evaluating system reliability in numerous industrial applications. Unfortunately, the required amount of testing for the individual components is typically not determined in a rigorous fashion with a system-level view. In practice, a simplistic approach that is commonly taken is to "allocate" the system reliability goal among constituent components and then conduct tests on the latter to meet the allocated component goal. This *ad hoc* approach completely ignores the specific system configuration. In other words, these component tests are not *system-based*. It can be demonstrated [Ea99] that when this naïve approach without explicit system-level objectives is used to draw inferences, it can lead to serious errors and unreliable conclusions.

We also wish to emphasize that the component testing approach absolutely does *not* imply the elimination of all system testing. Oftentimes component failures within a system may not be independent, so that tests on individual components or modules will not necessarily provide a completely accurate picture of system reliability. In such instances system level tests are essential. Indeed, in most situations at least some amount of system testing is always recommended (if not required) even if there is complete independence between component failures. However, component testing is still a superior option from an economic standpoint because it stands to reason that a sound program of testing at component levels can substantially reduce the amount of expensive system testing necessary at a later time. In such cases the issue to be addressed is the relative amounts of system and component level testing in a minimum cost plan.

2 System-Based Component Test Plans

In this section we overview the history and development of system-based component test plans and provide a generic problem formulation that holds across all test plans.

2.1 Background

The earliest work to address the topic of system-based component testing was a paper by Gal [Ga74], which considered an arbitrary coherent system of n different components with independent failures. In his plan, each component i is tested for t_i time units, and

the system is "accepted" if *no* failure occurs for each of the components during these prescribed testing periods. Suppose R_S denotes the system reliability for a unit time period (e.g., a suitably scaled version of the mission time) and c_i denotes the cost of testing component i for one unit of time. Given a value R_0 such that the system is considered *definitely unacceptable* if its reliability is below R_0, and a suitably low probability value β, Gal described a general procedure for obtaining the optimum test times that minimize the total testing cost $C(\underline{t}) = \sum_i c_i t_i$ subject to

$$P\{\text{"accept system" when } R_S \le R_0\} \le \beta. \quad [1]$$

He also gave some specific examples for series systems, parallel systems, and a serial connection of redundant units under the assumption that the component lifetimes are exponentially distributed.

Mazumdar [Ma77] extended this idea by considering a formulation identical to that of Gal but adding another probability constraint in keeping with standard statistical practice for determining the sample size. Given a value R_1 such that the system is considered *definitely acceptable* if its reliability is above R_1, and some suitably low probability value α, this probability constraint is:

$$P\{\text{" accept system" when } R_S \ge R_1\} \ge (1-\alpha). \quad [2]$$

These probability requirements are the same as those encountered in many conventional testing plans that are found for example, in the Department of the Navy document MIL-HDBK-781D [Mi87].

Since it is not always possible to satisfy both probability constraints [1] and [2] using Gal's criterion for system acceptance, a different criterion was needed. Mazumdar considered a *series* system with exponentially distributed component lifetimes and assumed that component testing took place with replacement. Suppose X_i denotes the number of failures of component i that occur when it is tested for t_i time units. He then proposed the following acceptance criterion: "accept the system if $\sum_i X_i \le m$," where m is a constant to be determined. He also proved that the optimum component test times are the same for each component irrespective of the testing costs, and that the optimum m is given by the smallest value for which both [1] and [2] are satisfied.

Much of the subsequent work that has been done builds on the basic framework advocated in the two seminal papers mentioned above. Numerous papers, including ones by Altinel [Al92, Al94], Altinel and Ozekici [AO97], Easterling et al. [Ea92], Mazumdar [Ma80], Raghavachari [Ra98], Rajgopal and Mazumdar [RM88, RM95, RM96, RM97], Vellaisamy and Sankar [VS02], Yan and Mazumdar [YM86, YM87a, YM87b], etc. have all considered the problem of designing component tests with different distributions for component lifetimes, different basic system configurations, different censoring schemes, and different amounts of *a priori* information about components. Systems with dependent components or with imperfect interfaces where a suitable combination of component and system tests may be required have not been considered as widely; some examples of work in this area include Altinel and Ozekici [AO98], Mazumdar and Rajgopal, [MR00], and Rajgopal, Mazumdar and Majety [RMM99]. Finally, there has also been some recent work that looks at other objectives such as minimizing the variance of the system reliability estimate [JC99].

With respect to software testing, the system-based component testing approach has not received much attention. While there is a wealth of literature on the use of operational profiles for software testing, such testing has always been addressed at the system level. Smidts and Sova [SS99] have noted some of the advantages from the use of "atomic" models in which the reliability assessment of the entire system is done based on the testing of its constituent modules or components. However, the only work that we are aware of that uses the general approach described earlier is some recent work by the authors [RM02].

2.2 A General Formulation for the Component Test Design Problem

To formulate the design of the component test plan as an optimization problem, suppose that the system has n different components and associated with the j^{th} component (j=1,2,…,n) is a parameter set Ω_j that determines its reliability. For example, with an exponentially distributed component lifetime, the component reliability is measured by the single parameter (say λ_j) representing its failure rate; so $\Omega_j \equiv \lambda_j$. It should be emphasized that the exact values of the parameters in this set are *unknown*. The system reliability R_S may be expressed as a function of Ω_1, Ω_2, etc., i.e., $R_S = R(\boldsymbol{\Omega})$, where (a) $\boldsymbol{\Omega}$ $=[\Omega_1, \Omega_2, ..., \Omega_n]$ is a collection of n parameter sets, and (b) the exact form of the function $R(\cdot)$ depends on the system configuration (series, parallel, serial connection of parallel systems, etc.). Let us define the two sets:

$$S_1 = \{\boldsymbol{\Omega} \mid R(\boldsymbol{\Omega}) \geq R_1\} \quad [3]$$
$$S_0 = \{\boldsymbol{\Omega} \mid R(\boldsymbol{\Omega}) \leq R_0\} \quad [4]$$

Thus S_1 is the set of *all* combinations of values for the *component* reliability parameters that lead to a *system* with a definitely acceptable reliability, while S_0 is the set of all values that leads to a system with a definitely unacceptable reliability. Suppose further that associated with each component j we define a function C_j that measures the cost of testing component j. Once again, the specific form of C_j will depend on the format followed by the test plan. The optimization problem of minimizing total test costs subject to constraints [1] and [2] may then be stated as follows:

Problem P: Minimize $Z = \sum_j C_j$

$$\text{st} \quad \text{Minimum}_{\boldsymbol{\Omega} \in S_1} \{P(\text{"Accept the system"})\} \geq 1-\alpha \quad [5]$$

$$\text{Maximum}_{\boldsymbol{\Omega} \in S_0} \{P(\text{"Accept the system"})\} \leq \beta \quad [6]$$

Note that values of R_1, R_0, α and β are typically specified for specific applications, (c.f., [Mi87]). Also note that the probability of accepting the system will in general be some function of the reliability of the components and how long we test them, i.e., of (i) the test-time vector $\boldsymbol{t}$, and (ii) the parameter set $\boldsymbol{\Omega}$. Problem P above is a two-stage optimization problem since each of its constraints is also an optimization problem.

- At the "inner" stage, assume that we are *given* a vector of test times $\underline{\boldsymbol{t}}$. Consider [2]: there may be many *different* $\boldsymbol{\Omega}$ in S_1 (each representing a different combination of component reliabilities) that lead to an acceptable system with $R_S \geq R_1$, and [2] requires that the probability of system acceptance should be at least $(1-\alpha)$ for *all* such $\boldsymbol{\Omega} \in S_1$. Equivalently, the *minimum* probability of acceptance across all $\boldsymbol{\Omega} \in S_1$ should be at least $(1-\alpha)$. This yields the first constraint of Problem P, namely [5], which imposes a restriction on the minimum probability of accepting an acceptable system. Along similar lines, the second constraint [6] imposes a restriction on the maximum probability of accepting an unacceptable system. Thus, given $\underline{\boldsymbol{t}}$, the LHS of [5] and [6] lead to two optimization subproblems (in $\boldsymbol{\Omega}$) over the sets S_1 and S_0 respectively. If the optimum values of these subproblems are respectively $\geq (1-\alpha)$ and $\leq \beta$ then the corresponding vector of test times $\underline{\boldsymbol{t}}$ is feasible in the main problem.
- The "outer" stage optimization problem is to find among all such feasible $\underline{\boldsymbol{t}}$ the particular vector that also minimizes the objective function.

2.2 More on the General Formulation

Problem P represents the optimization problem in its most general form. Specific test formats, system configurations, and failure time distributions of the components lead to specific definitions of the set $\boldsymbol{\Omega}$, the function $R(\boldsymbol{\Omega})$, the costs C_j, and the sets S_0 and S_1. In all cases though, observations on component failures must be combined into some *statistic*, which must then be incorporated into a *decision rule* for system acceptance. These in turn lead to specific mathematical translations of the probabilities in [5] and [6]. The choice of the best statistic and the corresponding decision rule are both open research issues. To be useful though, classes of rules must be limited to those that (a) make sense for the specific system under consideration, (b) offer insight into the system reliability, and (c) lead to analytically tractable formulations.

The discussion above demonstrates the complexity of the problem, from the statistical standpoint as well as the optimization standpoint; both of these aspects are closely tied together. From the statistical perspective the issues to address include the choice of the underlying failure time distributions, the relationship between component and system reliabilities, consideration of dependence, the choice of an appropriate test statistic and an appropriate acceptance rule, and the formulation of the optimization problems in terms of the distribution/test parameters. From the optimization perspective, the challenge is in developing algorithms to solve the optimization problems, deriving good approximations when the problem is otherwise intractable, and incorporating problem specific information into the solution methodology. An interesting feature of Problem P is that of the two optimization subproblems at the inner stage (minimization over S_1 and maximization over S_0) one problem typically turns out to be fairly easy, while the other is usually very hard to solve. That is, a test plan that satisfies one of producer's and consumer's risks is relatively easy to find, but simultaneously satisfying both is usually very challenging. For examples of different problem formulations and solution strategies the reader is referred to the references mentioned in the penultimate paragraph of Section 2.1.

3 An Application to Operational Testing of Software

It is a commonly recognized fact that software costs are far higher than those of hardware when considering the total cost of a computer-based system. Consequently, characterizing, measuring and evaluating software reliability have become crucial activities when developing large computerized systems ([He88], [MIO90]). Software testing models used in practice tend mostly to test the entire system as one unit without paying particular attention to its individual component modules and the logical connections that exist among them. However, researchers have begun to advocate the use of "atomic" models in which the entire system is evaluated based on the testing of its constituent modules or components, since such models more readily allow for reliability assessment and determination of optimal testing strategies [SS99]. As an example, different modules of complex software systems are often developed simultaneously by independent teams and then integrated. As another example one might have commercial-off-the-shelf (COTS) components that are used in different software systems or versions. In both cases, if these modules are tested on an individual basis then explicit consideration must be given to how they will fit together in the overall system. In this section we describe an application of our approach of system-based component testing to the area of *operational testing* of such components. In this testing mode, instead of actively looking for failures, one waits for the failures to surface through repeated testing using random samples from the hypothesized statistical distribution of the input values. The reader is referred to [Fr98] for a further discussion of this subject.

3.1 Representation of System Reliability

A challenge in using a component testing approach is that software systems do not necessarily have series/parallel structures and one must develop other ways to define system reliability in terms of module reliabilities. In [KMM99] the authors use a Bayesian approach to do this. We take a different approach based on classical statistics. As proposed in [PMM93], we first define a software *system* as a "collection of programs and system files such that the system files are accessed and altered only by the programs in the collection." Each element in this collection (e.g., program, subprogram, file) is called a *module*. The performance (and hence the reliability) of the system depends upon (1) the reliability of each of the constituent modules, and (2) the relationship between these modules in the context of the system. In this regard a software system is exactly like any other type of system. However, unlike other systems the actual relationship between system and module reliabilities is quite unique and in addition to the system structure, it also depends upon the specific definition of software reliability.

It is possible to view software reliability from the standpoint of a mission time. However, with the emphasis on reusable software we view it from the more common perspective of general use on a variety of different inputs. In this case, the reliability is simply defined as the probability that the system will correctly process a randomly chosen input. Given that the precise nature of the applications on which some general-purpose software system will be used is not known in advance, software use is usually quantified by defining a suitable distribution or *operational profile* [MIO90]. Essentially,

one may view this as a description of all possible inputs to the software system along with the relative frequency with which each input might be encountered. The operational profile and the program structure allow us to develop a statistical distribution of inputs for each individual module and in the modular testing approach, inputs are randomly selected from this distribution.

One particular characterization for relating system and module reliabilities is through the use of Markov chains ([Li75], [Ch80], [MLW95]). Specifically, we make use of Cheung's model [Ch80], which is based on the assumption that the transfer of control between individual modules of a software system takes place according to a Markov chain; this model has been used elsewhere as well ([Si88], [PMM93]). In essence, the probability p_{ij} that control transfers from module i to another module j is independent of how module i was entered. First, consider a perfectly reliable system with n modules (components), where module 1 represents the initial state (e.g., this could be the main program called by the user). Also define an additional "terminal" module S to which control is transferred upon successful program completion (e.g., S might be the operating system) with probability p_{iS} of being entered from state i. Note that for all i we have $p_{iS} + \sum_{j=1}^{n} p_{ij} = 1.$

Since we consider systems that are not 100% reliable, suppose that module i is imperfect and has reliability r_i, i.e., P(the module i does not fail any time that control enters it) = r_i. To model this (imperfect) system, define an additional state F to represent program failure. Since each module is imperfect F can be entered from any module (except S). Note that unlike the transient states 1,2,…,n, the states F and S are absorbing states and represent (respectively) program failure and successful program completion. The Markov chain thus has n+2 states (1,2,…,n,F,S) and a transition matrix Q with transition probabilities given by $q_{FF}=q_{SS}=1$, $q_{iF} = (1-r_i)$ for $i=1,2,\ldots,n$; $q_{ij} = r_i p_{ij}$ for $i=1,2,\ldots,n$ and $j=1,2,\ldots,n,S$; and all other $q_{ij}=0$. Note that q_{ij} captures the probability that module i does not fail and successfully transfers control to module j. For a 4-component system the transition matrix Q is as shown below:

$$Q = \begin{bmatrix} r_1p_{11} & r_1p_{12} & r_1p_{13} & r_1p_{14} & r_1p_{1S} & 1-r_1 \\ r_2p_{21} & r_2p_{22} & r_2p_{23} & r_2p_{24} & r_2p_{2S} & 1-r_2 \\ r_3p_{31} & r_3p_{32} & r_3p_{33} & r_3p_{34} & r_3p_{3S} & 1-r_3 \\ r_4p_{41} & r_4p_{42} & r_4p_{43} & r_4p_{44} & r_4p_{4S} & 1-r_4 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ S \\ F \end{matrix}$$
$$\begin{matrix} 1 & 2 & 3 & 4 & S & F \end{matrix}$$

Assuming that state 1 represents the initial state (i.e., control is initiated by module 1), one may use a standard method of computing absorption probabilities [Ci75], to show that the reliability R_S of the system can be computed via

$$R_S = \sum_{i=1}^{n} [I_n - \hat{Q}]^{-1}_{1i} r_i p_{iS} \qquad [7]$$

where I_n is the identity matrix of order n, and $\hat{Q}$ is the ($n \times n$) submatrix of Q obtained by dropping its last two rows and columns. In other words, the reliability of the system is the inner product of the first row of the inverse of the matrix ($I_n - \hat{Q}$) and the column of $\hat{Q}$ corresponding to state S.

Just as we assumed in Section 2.2 that the component reliability parameters given by Ω_i were unknown, we assume here that the values of all r_i are unknown (indeed, if these were known there would be no reason to do any testing). However, we assume that the p_{ij} are known/computed based on the operational profile; recall that the operational profile provides the probability of encountering a specific type of input, and the logic of the software provides the specific path of modules through which control is transferred for that input. Thus one may compute the probability that control transfers from module i to module j for any input from the use distribution. Clearly, this probability depends on the operational profile. Thus one must pay special attention to the development of the operational profile, and a caveat that we offer is that the proposed approach is accurate only as long as the operational profile is accurate. However, this is generally true whenever any statistical approach is used to test software and this point is also discussed at length in [MIO90]. If the operational profile changes over time one should compute new probabilities p_{ij} for the new profile. However, the formulae used to compute these would be identical to the ones used initially as long as the software itself is not altered.

3.2 Formulation of an Optimum System-Based Component Test Plan Model

In our test plan, we run k_i unit tests on module i where each test uses inputs drawn at random from the use distribution for the module. Suppose further that we observe X_i failures among the k_i test instances of module i. It is assumed that $X_1, X_2, \ldots, X_n$ are mutually independent random variables. The following rule is then used for the overall system: "Accept the system as reliable as long as $\Sigma_i X_i = 0$." Note that this test plan is virtually identical to the one used in Gal's original paper on component testing [Ga73]. Similar to Gal's work we only consider Type II error, i.e., we wish to ensure that the probability of the plan accepting a system with reliability $R_S \leq R_0$ is smaller than some specified small fraction β, where R_0 is a prespecified value and R_S follows from [7]. In optimizing the design of this test plan the objective is minimize test effort by finding the smallest value for each k_i that provides us with this level of protection from Type II error. Note that with inputs drawn at random from the use distribution, the proposed test format effectively subjects each module to a series of independent functional tests with inputs representative of the frequency of use in the final operational environment.

Let $\boldsymbol{r} \in R^n$ denote the (unknown) module reliabilities, i.e., $\boldsymbol{r} = [r_1, r_2, \ldots, r_n]$. Clearly, X_i has a binomial distribution with parameters k_i and $(1-r_i)$ so that $E[X_i]=k_i(1-r_i)$. Since k_i is likely to be large and $(1-r_i)$ to be small, we can use a Poisson approximation for the distribution of X_i. The fact that the tests on the modules are independent of each other then implies that $\sum_i X_i$ is also approximately Poisson with parameter $\sum_i k_i(1-r_i)$. Thus the

probability of system acceptance = $P(\sum_i X_i=0) = \exp[-(\sum_i k_i(1-r_i)]$. Finally, (along the lines of the set S_0 that we defined via [4]) let us define

$$Q_0 = \{ \boldsymbol{r} \in R^n \mid \boldsymbol{0} \le \boldsymbol{r} \le \boldsymbol{1},\ R_S(\boldsymbol{r}) \le R_0 \}, \qquad [8]$$

where $R_S(\boldsymbol{r})$ is the reliability of a system whose module reliabilities are given by the vector $\boldsymbol{r}$ and is computed using the formula in [7]. Note that (like the set S_0 in [4]) Q_0 denotes the set of all module reliability vectors for which the system is definitely unreliable. In general, Q_0 could have infinitely many elements. The problem of designing a test plan may now be stated as the following mathematical program:

$$\begin{aligned} &\text{Minimize } (k_1+k_2+\ldots+k_n) \\ &\text{subject to } e^{\left(-\sum_i k_i(1-r_i)\right)} \le \beta \text{ for all } \boldsymbol{r} \in Q_0 \\ &\qquad k_i \ge 0 \text{ and integer for } i=1,2,\ldots,n. \end{aligned} \qquad [9]$$

Note that the above problem is an integer linear program in k_i with infinitely many constraints (one for each member of Q_0). In order to solve the problem we develop an inner-stage optimization problem to replace the constraint given by [9]. This is exactly the same as our general approach described in Secton 2.2 (where we used [6] to replace the basic constraint given by [1]). Since $\exp[-(\sum_i k_i(1-r_i)] \le \beta$ is equivalent to $\sum_i k_i(1-r_i)] \ge -ln(\beta)$ it follows that the above optimization problem may be rewritten as follows:

$$\begin{aligned} &\text{Minimize } k_1+k_2+\ldots+k_n \\ &\qquad \text{st} \quad \{\text{Minimum } \Sigma_i\, k_i(1-r_i), \text{ st } \boldsymbol{r} \in Q_0\} \ge -ln\,(\beta) \\ &\qquad\qquad k_i \ge 0 \text{ and integer for } i=1,2,\ldots,n. \end{aligned} \qquad [10]$$

Again, note that as in Section 2.2 we have a two-stage mathematical program:

- In the "inner" stage we assume we are given a vector $\boldsymbol{k}=[k_1, k_2,\ldots, k_n]$ and solve a problem in the r_i; this problem has a linear objective and one nonlinear constraint (corresponding to $R_S(\boldsymbol{r}) \le R_0$) in addition to simple upper and lower bounds on the r_i. If the resulting optimum objective value exceeds $-ln\,(\beta)$, then $\boldsymbol{k}$ is feasible.
- In the "outer" problem the objective is to find among all feasible $\boldsymbol{k}$ the particular vector that minimizes $\Sigma_i k_i$.

3.3 A Cutting-Plane Algorithm to Solve the Problem

As a matter of simplicity we ignore the integer restrictions on k_i and solve a continuous problem. If the optimum values are rounded the solution will obviously remain feasible, and as long as the k_i are large, deviations from the optimum will be negligible. The procedure is as follows:

STEP 0: Choose a set of elements from the set Q_0 - say n vectors $\boldsymbol{r^1}, \boldsymbol{r^2},\ldots,\boldsymbol{r^n}$ - and use these to define an initial set of constraints in [9] – note that these constraints will be in terms of the variables $k_1,k_2,\ldots,k_n$.. Solve the resulting linear program where the objective is to minimize $\Sigma_j k_j$ subject to these n constraints and $k_i \ge 0$. Call the solution $\boldsymbol{k}$.

STEP 1: Use the vector $\boldsymbol{k}$ to define the (nonlinear) optimization problem in r_i as given by the LHS of [10]. Solve this second-stage optimization problem and let the optimum solution be $\boldsymbol{r}=[r_1, r_2, \ldots, r_n]$. If $\Sigma_i\ k_i(1-r_i) \geq -ln\ (\beta)$ then stop; the current vector $\boldsymbol{k}$ is optimal. Otherwise proceed to Step 2.

STEP 2: Redefine the current linear program with the extra constraint $\Sigma_i\ k_i(1-r_i) \geq -ln\ (\beta)$, where the r_i come from Step 1. Clearly, this cuts out the current solution vector $\boldsymbol{k}$, which is now infeasible. Solve the linear program and obtain a new solution vector and replace $\boldsymbol{k}$ with this vector. Then go to Step 1.

3.4 A Numerical Example

To illustrate the approach, we present the system pictured in the figure below which was motivated by a radar software system discussed in [SR84].

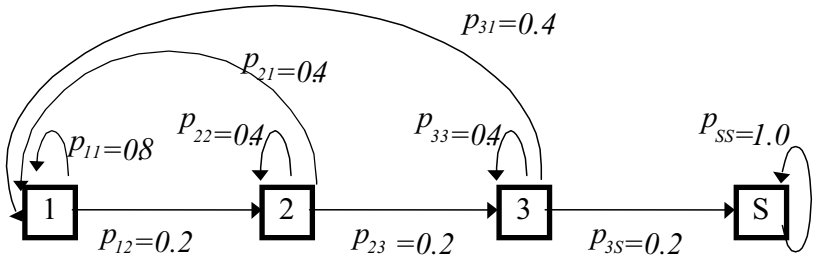

Figure 1: A Modular Software System

Suppose that a value of 0.95 is specified for R_0 and a value of 0.05 for β. The reliability R_S for this system derived using the procedure in Section 2 yields

$$R_S(r_1,r_2,r_3) = \frac{0.008 r_1 r_2 r_3}{(1-0.8r_1)(1-0.4r_2)(1-0.4r_3) - 0.08 r_1 r_2 (1-0.4r_3) - 0.016 r_1 r_2 r_3}$$

Note that in general [7] requires the use of a system capable of symbolic algebra (such as *Maple*® or *Mathematica*®) for nontrivial problems. The constraint $R_S(\boldsymbol{r}) \leq 0.95$ for the inner-stage problem yields the following after some algebraic manipulation:

$$0.8r_1 + 0.4r_2 + 0.4r_3 - 0.24r_1r_2 - 0.32r_1r_3 - 0.16r_2r_3 + [0.112+(0.008/0.95)]r_1r_2r_3 \leq 1.$$

We thus minimize $k_1(1-r_1) + k_2(1-r_2) + k_3(1-r_3)$ subject to the above constraint and the restriction that $r_1, r_2, r_3 \in [0,1]$. The algorithm converged to the optimum solution $\boldsymbol{k}$ = [2564.6, 857.2, 288.8]. Rounding up, we would thus test 2,565 random inputs on the first module, 858 on the second, and 289 on the third, and the system would be accepted as reliable as long as we observe no failures from these tests.

4 Conclusions

This paper extends the idea of optimum, system-based component testing from the general domain to software reliability evaluation. This is done in the context of a specific Markov model for software reliability. A model and procedure are described for finding the minimum number of test cases required for the different modules of a software system while ensuring that there is no more than some small prespecified probability of accepting a system whose reliability is below a prescribed lower bound. It is worth mentioning that if the cost of testing were different from one module to the other, this would be easy to incorporate into our solution procedure. The only change would be in the objective function of the linear program, which would now be $\Sigma_i c_i k_i$ instead of $\Sigma_i k_i$ where c_i is the test cost for module i. There are numerous opportunities for future work such as more extensive testing of this procedure on large-scale software, incorporation of Type 1 error bounds, evaluation of other possible software reliability models, and the use of other testing regimens.

Bibliography

[Al92] Altinel, I.K.: "The Design of Optimum Component Test Plans in the Demonstration of a Series System Reliability," *Computational Statistics and Data Analysis*, **14**, 1992, pp. 281-292.

[Al94] Altinel, I.K.: "The Design of Optimum Component Test Plans in the Demonstration of System Reliability," *European Journal of Operations Research*, **78**, 1994, pp. 97-115.

[AO97] Altinel, I.K.; Ozekici, S.: "A Dynamic Model for Component Testing," *Naval Research Logistics*, **44**, 1997, pp. 187-197.

[AO98] Altinel, I.K.; Ozekici, S.: "Optimum Component Test Plans for Systems with Dependent Components," *European Journal of Operations Research*, **111**, 1998, pp. 175-186.

[BP75] Barlow, R.E.; Proschan, F.: *Statistical Theory of Reliability and Life Testing*, New York: Holt, Rinehart and Winston, 1975.

[Ch80] Cheung, R. C.: "A User-Oriented Software Reliability Model," *IEEE Transactions on Software Engineering*, **SE-6**, 1980, pp. 118-125.

[Ci75] Cinlar, E.: *Introduction to Stochastic Processes*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

[Ea91] Easterling, R.G.; Mazumdar, M.; Spencer, F.W.; Diegert, K.V.: "System Based Component Test Plans and Operating Characteristics: Binomial Data," *Technometrics*, **33**, 1992, pp. 287-298.

[Fr98] Frankl, P.G.; Hamlet, R.G.; Littlewood B.; Strigini, L.: "Evaluating Testing Methods by Delivered Reliability," *IEEE Transactions on Software Engineering*, **24**, 1998, pp. 586-601.

[Ha74] Gal, S.: "Optimal Test Design for Reliability Demonstration," *Operations Research*, **22**, 1974, pp. 1236-1242.

[He88] Hetzel, W.C.: *Complete Guide to Software Testing*, QED Information Sciences, Inc., Wellesley, MA, 1988.

[JC99] Jin, T.; Coit, D.: "Allocation of Test Units to Minimize System Reliability Estimation Variability," Rutgers University Industrial Engineering Department, Working Paper 99-122, 1999.

[KMH99] Kuball, S.; May, J.; Hughes, G.: "Building a System Failure Rate Estimator by Identifying Component Failure Rates," *Proceedings of the Tenth International Symposium on Software Reliability Engineering*, 1999, pp. 32-41.

[Li75] Littlewood, B.: "A Reliability Model for Systems with Markov Structure," *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, **24**, 1975, pp. 172-177.

[MSS74] Mann, N.R.; Schafer, R.E.; Singpurwalla, N.D.:, *Methods for Statistical Analysis and Life Data*, New York: John Wiley, 1974.

[Ma77] Mazumdar, M.: "An Optimum Procedure for Component Testing in the Demonstration of Series System Reliability," *IEEE Transactions on Reliability*, **R-26**, 1977, pp. 342-345.

[Ma80] Mazumdar, M.: "An Optimum Component Testing Procedure for a Series System with Redundant Subsystems, *Technometrics*, **22**, 1980, pp. 23-27.

[MR00] Mazumdar, M.; Rajgopal, J.: "Minimum Cost Test Plans for a Series System with Imperfect Interfaces," in *Perspectives in Statistical Science*, (Basu, A.K., Ghosh, J.K., Sen, P.K. and Sinha, B.K., Eds.), Oxford University Press, New Delhi, 2000.

[MLW95] Meyer, J. F.; Littlewood, B.; Wright, D. R.:"Dependability of Modular Software in a Multiuser Operational Environment," *Proceedings of the Sixth International Symposium on Software Reliability Engineering*, 1995, pp. 170-179.

[Mi87] MIL-HDBK-781: Washington: Department of the Navy, Space and Naval Warfare Systems Command, Washington DC 20363, July 14, 1987.

[MIO90] Musa, J. D.; Iannino A.; Okumoto, K.: *Software Reliability*, McGraw-Hill Publishing Co., N.Y., 1990.

[PMM93] Poore, J. H.; Mills, H. D.; Mutchler, D.:"Planning and Certifying Software Systems Reliability," *IEEE Software*, 1993, pp. 88-99.

[Ra98] Raghavachari, M.: "A Note on Optimal Component Test Plans for Series System Reliability With Exponential Failure Times," *Technometrics*, **40**, 1998, pp. 345-347.

[RM98] Rajgopal, J.; Mazumdar, M.: "A Type-II Censored, Log Test-Time Based Component Testing Procedure for a Parallel System," *IEEE Transactions on Reliability*, **37**, 1988, pp. 406-412.

[RM95] Rajgopal, J.; Mazumdar, M.: "Designing Component Test Plans for Series System Reliability via Mathematical Programming", *Technometrics*, **37**, 1995, pp. 195-212.

[RM96] Rajgopal, J.; Mazumdar, M.: "A System Based Component Test Plan for a Series System, with Type-II Censoring," *IEEE Transactions on Reliability*, **45**(3), 1996, pp. 375-378.

[RM97] Rajgopal, J.; Mazumdar, M.: "Minimum Cost Component Test Plans for Demonstrating Reliability of a Parallel System," *Naval Research Logistics*, **44**, 1997, pp. 401-418,.

[RM02] Rajgopal, J.; Mazumdar, M.:" Modular Operational Test Plans for Inferences on Software Reliability Based on a Markov Model," *IEEE Transactions on Software Engineering*, **28(4)**, 2002, pp. 358-363.

[RMM99] Rajgopal, J.; Mazumdar, M.; Majety, S.V.: "Optimum Combined Test Plans for Systems and Components," *IIE Transactions*, **31(6)**, 1999, pp. 481-490.

[Si88] Siegrist, K.: "Reliability of Systems with Markov Transfer of Control. *IEEE Transactions on Software Engineering*, **14**, 1988, pp. 1049-1053.

[SS99] Smidts and D. Sova, "An Architectural Model for Software Reliability Quantification: Sources of Data," *Reliability Engineering and System Safety*, **64**, 1999, pp. 279-290.

[SR84] Soistman, E.C.; Ragsdale, K.B.: Combined Hardware/Software Reliability Prediction Methodology. Rome Air Development Center Contract Report OR-18-173, II,, 1984.

[VS02] Vellaisamy, P.; Sankar, S.: "Two-Stage Component Test Plans for Testing the Reliability of a Series System," *Naval Research Logistics*, **49(1)**, 2002, pp. 95-116.

[YM86] Yan, J.H.; Mazumdar, M.: "A Comparison of Several Component Testing Plans for a Series System," *IEEE Transactions on Reliability*, **R-35**, 1986, pp. 437-443.

[YM87a] Yan, J.H.; Mazumdar, M.: "A Component Testing Plan for a Parallel System with Type II Censoring," *IEEE Transactions on Reliability*, **R-36**, 1987, pp. 425-428.

[YM87b] Yan, J.H.; Mazumdar, M.: "A Comparison of Several Component Testing Plans for a Parallel System," *IEEE Transactions on Reliability*, **R-36**, 1987, pp. 419-424.