

Augmenting system tests with component tests for reliability assurance

Richard L. Warr, Jace Ritchie & Michael Hamada

To cite this article: Richard L. Warr, Jace Ritchie & Michael Hamada (24 Jan 2024): Augmenting system tests with component tests for reliability assurance, Journal of Quality Technology, DOI: [10.1080/00224065.2023.2287747](https://doi.org/10.1080/00224065.2023.2287747)

To link to this article: <https://doi.org/10.1080/00224065.2023.2287747>



Published online: 24 Jan 2024.



Submit your article to this journal [↗](#)



Article views: 38



View related articles [↗](#)



View Crossmark data [↗](#)



Augmenting system tests with component tests for reliability assurance

Richard L. Warr^a , Jace Ritchie^b , and Michael Hamada^c 

^aDepartment of Statistics, Brigham Young University, Provo, Utah; ^bMathematics and Statistics Department, Utah State University, Logan, Utah; ^cStatistical Sciences Group, Los Alamos National Laboratory, Los Alamos, New Mexico

ABSTRACT

To successfully determine if a system meets some reliability standard, typically assurance tests are conducted on the full system. This proven approach is extremely effective. However, it can be cost prohibitive when full system tests are expensive. Previous work has incorporated component and subsystem data to supplement the component prior distributions before a full system assurance test. Although this can be helpful, we propose an additional advance of allowing component and subsystem tests as part of the assurance test. We show that by augmenting full system tests with component and subsystem tests, in some cases, we can reduce the cost of the assurance test. For a system that can be decomposed into independent series and parallel parts, the theory works perfectly. In the case where we cannot accurately classify parts using the series or parallel structure, we add a generalization that works for systems that are “almost-series” or “almost-parallel.”

KEYWORDS

Bayesian testing;
component test; consumer's
risk; cost reduction;
demonstration test;
producer's risk

1. Introduction

Assurance testing is an important part of product development. It balances both the interest of the producer as well as the interest of the consumer. Both parties benefit with a successful system, but are also invested in diverging goals. The producer wants the system to pass the assurance test to gain the monetary payout of successfully fulfilling a contract. However, even if the system meets specifications, there is a chance (i.e., probability) that the assurance test will fail; this is the producer's risk which should be controlled. Conversely, the consumer wants to ensure the system meets specifications so that they have a product that is useful and reliable. The risk for the consumer is the chance that the system does not meet specifications but passes the assurance test. As with the producer's risk, the consumer's risk should also be controlled.

A viable assurance test is one that meets both the required producer's and consumer's risks. This typically requires a specific number of system tests (n), balanced with an acceptable number of failures (c). Typically in assurance testing, the binomial distribution is assumed with n total system tests and c acceptable failures. If, after n tests with c or fewer failures occurring, the assurance test is passed, otherwise the assurance test is failed. To find an assurance test that

appropriately balances the aforementioned risks, generally n or c is increased to reduce one or both of the probabilities. In practice, n is minimized to reduce the cost of the assurance test without exceeding the consumer's and producer's risk thresholds.

Meeker and Escoba (2004) make an important distinction between demonstration and assurance testing, mainly that demonstration testing relies entirely on the outcome of the test. Essentially, demonstration testing follows the frequentist paradigm, and assurance testing allows for other information to be taken into account and is usually approached using Bayesian methodology. However, demonstration testing ignores the information gathered by previous tests on similar systems with similar components. Bayesian assurance testing provides a methodology to control both types of risk using prior information updated by system tests. This prior information is most useful if it reduces the total number of system tests needed to find an acceptable assurance test. However, the prior probabilities and the data used to update them are often based entirely on system tests. Full system tests are generally more expensive than component tests (e.g., testing if a rocket works can be more expensive than testing the engine, navigating system, and other subsystems independently). Hence, a framework that uses component tests to augment the system tests can

be very advantageous in reducing the overall cost of testing.

It is often useful to think of systems in terms of series or parallel structures. This provides a framework from which component reliability is easily incorporated into system reliability. Indeed, system reliability can be described completely by the components, provided that components in the system are independent. Hence, an assurance test can be found for any such system using only component tests. In this case, augmenting system tests with component tests is straightforward since a test on the system is roughly equivalent to a test on each component. Thus, the assurance test need only be restricted by the expenses associated with component and system tests.

However in reality, there are systems in which the components are not independent or in which the series or parallel structures do not accurately describe the reliability of the system as a whole. For example, consider a system composed of a series of components whose reliability is not exactly the product of the component reliabilities. In such a case, component tests alone are not sufficient to find an assurance test for the system since the assumed relationships between components do not hold. These situations are fairly typical of real-world systems. Consider a braking system, which is composed of brake pads and the mechanism that presses the pads to a wheel. As the brake pads start to wear down, the mechanism must apply more pressure to the brakes to stop the wheel, thus increasing its own risk of failure. Hence, the probability of failure is greater than just the probabilities of failure of the components multiplied together. More complex systems can have many such interactions between components. Therefore, we add a generalization that permits using component assurance tests to augment the system assurance test data without assuming that component relationships hold exactly.

The remainder of this article is organized as follows. In the next section we discuss the existing literature related to assurance tests. Then we introduce our proposed methodology of using component tests to augment a Bayesian assurance test first in a simple series system scenario and then finish that section with the most general case. In the next section, we demonstrate the performance of our method with a few simulation studies. Finally, our method is implemented on a complex system, which is followed by a discussion of our work and possible future areas of research.

2. Literature review

In this article, we consider one-shot systems that either work or do not work when used. That is, a system test results in a pass or fail. We also focus specifically on Bayesian assurance tests, which can incorporate any relevant and supplementary data or subject-matter knowledge into a prior distribution using Bayesian inference. The simplest case is where system tests are viewed as Bernoulli trials and prior information about the probability a trial results in a pass is incorporated into a beta prior distribution (Hamada et al. 2008). In a more complex model, Hamada et al. (2008) also presents a model with a hierarchical prior distribution. That is, the system to be tested is assumed to come from a population of systems whose reliabilities follow a distribution, and previous test data are available from a sample of these systems. Wilson and Farrow (2021) also use the simple beta and hierarchical prior distributions in developing system assurance tests. Hamada et al. (2014) considers system assurance tests where the system reliability prior distribution is developed from the system reliability model based on the component reliabilities and available and relevant component test data. Hamada et al. (2014) suggests that the components may be similar but not identical to those of the system to be tested. Wilson and Fronczyk (2017) present system assurance tests and refer to Hamada et al. (2008) and Hamada et al. (2014) for more details. In addition to component test data, Li et al. (2017) considers available and relevant subsystem test data to develop the system reliability prior distribution.

Hamada et al. (2008, 2014) consider consumer's and producer's risk-based assurance tests. The posterior consumer's risk is the posterior probability that the reliability is no more than a specified rejectable reliability level π_R^* given the assurance test is passed. The posterior producer's risk is the posterior probability that the reliability is at least a specified acceptable reliability level π_A^* given the assurance test is failed. The consumer's and producer's risk-based assurance tests ensure that the consumer's and producer's risks are small and no larger than prespecified probabilities α and β , respectively.

Ten and Xie (1998) and Guo, Jin, and Mettas (2011) also present system assurance tests but use a prior distribution approximated by a beta distribution based on relevant component data or information, whereas Hamada et al. (2008, 2014) use the exact system prior distribution. Additionally, Ten and Xie (1998) and Guo, Jin, and Mettas (2011) only consider consumer's risk-based assurance tests.

Smith, Kelly, and Dezfuli (2010) considers component assurance tests that maximize the median of the system reliability posterior distribution subject to an overall cost constraint for specified test costs for the different components in the system. Smith, Kelly, and Dezfuli (2010) presents zero-failure component assurance tests, i.e., the assurance test passes if there are no failures in the component tests, and assumes that the system reliability model holds to obtain the system posterior distribution from the resulting component posterior distributions.

In this article, we consider assurance tests with both system and component tests. We also do not assume that the system reliability model based on the component reliabilities alone holds exactly. For example, a series system reliability may be less than the product of the component reliabilities. We also consider assurance tests which satisfy both consumer's and producer's risk thresholds.

3. Methodology

In this section, we show how to use component assurance tests to augment system assurance tests. We first start with a simple series system, and then show how the method can be generalized to more complex situations.

3.1. Series systems

Consider a system with k components in series as shown in Figure 1. Each component's prior information (previous test data, expert judgment, simulation studies, etc.) is contained in its respective prior distribution $p(\pi_1), \dots, p(\pi_k)$, where π_i is the probability that the i th component works when tested. Together these k components combine to determine the system parameter, π_s , as defined by

$$\pi_s = \pi_1 \times \dots \times \pi_k, \quad [1]$$

which assumes independent components. For a general system we use the notation $\pi_s = f(\boldsymbol{\pi})$, where $f(\cdot)$ is a function that encodes the system structure, and $\boldsymbol{\pi} = (\pi_1, \dots, \pi_k)^T$. Thus for a given $f(\cdot)$ and $\boldsymbol{\pi}$, π_s is determined.

The system prior distribution $p(\pi_s)$ is induced by the component prior distributions. Since we are interested in draws from the system prior we do not

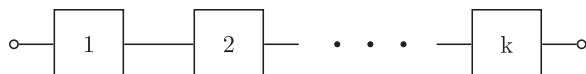


Figure 1. A k -component series system.

obtain a closed form distribution. For a series system, with independent components, we use draws from the component prior distributions and multiply them according to Eq. [1] to obtain draws from the system prior distribution. Note that the component prior distributions need not be independent as long as draws from their joint prior distribution can be taken.

Now suppose that an assurance test consists of n_s system tests and n_1, \dots, n_k components tests. We define X_s to be the number of successes for the system tests and X_i (for $i \in \{1, 2, \dots, k\}$) to be the number of successes for the i th component tests.

The number of acceptable failures for each component is denoted as c_i for the i th component and let c_s be the number of acceptable failures for the system tests. Thus, if $X_i \geq n_i - c_i$ for all $i \in \{1, 2, \dots, k\}$ and $X_s \geq n_s - c_s$, then we say the assurance “test is passed” or the event T_P occurred. If T_P did not occur, then we say the assurance “test is failed” and the event T_F occurred. It logically follows that $P(T_P) = 1 - P(T_F)$.

At a more granular level, we define T_{P_i} or T_{F_i} if the test is passed or the test is failed for the i th component, respectively. These can be computed by summing the appropriate binomial probabilities. For example, for the i th component,

$$P(T_{P_i} | \pi_i) = \sum_{x_i=n_i-c_i}^{n_i} \binom{n_i}{x_i} \pi_i^{x_i} (1 - \pi_i)^{n_i-x_i}.$$

Additionally, for the n_s system tests, we define the events T_{P_s} or T_{F_s} . Therefore, the event that the aggregate assurance test is passed is defined as

$$T_P = T_{P_s} \cap T_{P_1} \cap \dots \cap T_{P_k}.$$

This pass/fail criterion is not the only possible choice, and is somewhat conservative. Other criteria might be better able to compensate between varied performance of individual components. For example, an overall system pass might be warranted, if one component barely fails and all other components easily pass.

Finally, we define two quantities that determine reliability levels. The first is the acceptable reliability level denoted as π_A^* . The other is the rejectable reliability level which we specify as π_R^* , such that $\pi_R^* \leq \pi_A^*$. The objective is to find an assurance test that satisfies both

$$P(\pi_s \geq \pi_A^* | T_F) \leq \alpha \quad [2]$$

and

$$P(\pi_s \leq \pi_R^* | T_P) \leq \beta \quad [3]$$

simultaneously, where α and β are the maximum acceptable probability thresholds for the producer's and consumer's risk, respectively.

Although we desire an assurance test that satisfies Eqs. [2] and [3], there are an infinite number of solutions. To narrow it down to one solution, we want to select the test that optimizes some preestablished criterion, which could be the total number of tests, test costs, time on test, or some other factor (or combinations of factors). Once the optimization criterion is set, the solution is determined by selecting various values for n_1, n_2, \dots, n_k and n_s and determining which has optimal cost, while satisfying Eqs. [2] and [3].

Now we show the formulas for $P(\pi_s \geq \pi_A^* | T_F)$ and $P(\pi_s \leq \pi_R^* | T_P)$. Since both T_P and T_F are events defined by the assurance test data, $P(\pi_s \geq \pi_A^* | T_F)$ and $P(\pi_s \leq \pi_R^* | T_P)$ are posterior quantities. They can be found using the following formulae:

The general form of the posterior producer's risk (PPR) equation is:

$$P(\pi_s \geq \pi_A^* | T_F) = \frac{\int_0^1 \dots \int_0^1 I(\pi_s \geq \pi_A^*) [1 - P(T_P | \boldsymbol{\pi})] p(\pi_s) d\pi_1 \dots d\pi_k}{\int_0^1 \dots \int_0^1 [1 - P(T_P | \boldsymbol{\pi})] p(\pi_s) d\pi_1 \dots d\pi_k}, \quad [4]$$

where for a series system

$$P(T_P | \boldsymbol{\pi}) = P(T_P | \pi_s) \prod_{i=1}^k P(T_P | \pi_i). \quad [5]$$

We note that Eqs. [4] and [5] are defined in terms of T_P events rather than T_F events. This is done for convenience, as there is only one way the aggregate test is passed but many possible ways the aggregate test is failed. The general form of the posterior consumer's risk (PCR) is:

$$P(\pi_s \leq \pi_R^* | T_P) = \frac{\int_0^1 \dots \int_0^1 I(\pi_s \leq \pi_R^*) P(T_P | \boldsymbol{\pi}) p(\pi_s) d\pi_1 \dots d\pi_k}{\int_0^1 \dots \int_0^1 P(T_P | \boldsymbol{\pi}) p(\pi_s) d\pi_1 \dots d\pi_k}, \quad [6]$$

where, for a series system, $P(T_P | \boldsymbol{\pi})$ is expressed in Eq. [5].

The fact that these formulas contain multidimensional integrals makes the analytic computation of these quantities quite difficult. In this article, we adopt a Monte Carlo computational approach to estimate the producer's and consumer's risks in Eqs. [4] and [6], respectively.

Calculating the posterior probabilities of interest in Eqs. [4] and [6] are key to implementing our proposed method. A standard approach for computing intractable integrals is Monte Carlo estimation; and we refer the interested reader to Robert and Casella

(2004) for a detailed text on Monte Carlo estimation methods. Next, we provide formulas to use Monte Carlo estimation to obtain the posterior consumer's and producer's risk.

For Eq. [4], a Monte Carlo approximation of the posterior producer's risk can be calculated by obtaining a large sample (of size m) from each prior contained in the system.

In the k-series system this is just a sample from each $p(\pi_i)$ for $i = 1, \dots, k$. Let $\pi_i^{(j)}$ be the j^{th} sample (for $j \in \{1, 2, \dots, m\}$) from the prior of the i^{th} component, $p(\pi_i)$. Note that the j^{th} sample from the system prior is $\pi_s^{(j)} = \prod_{i=1}^k \pi_i^{(j)}$ and

$$P(T_P | \boldsymbol{\pi}^{(j)}) = P(T_P | \pi_s^{(j)}) \prod_{i=1}^k P(T_P | \pi_i^{(j)}). \quad [7]$$

The Monte Carlo approximation to Eq. [4] is:

$$P(\pi_s \geq \pi_A^* | T_F) \approx \frac{\frac{1}{m} \sum_{j=1}^m I(\pi_s^{(j)} \geq \pi_A^*) (1 - P(T_P | \boldsymbol{\pi}^{(j)}))}{1 - \frac{1}{m} \sum_{j=1}^m P(T_P | \boldsymbol{\pi}^{(j)})}. \quad [8]$$

Similarly, for Eq. [6] we have

$$P(\pi_s \leq \pi_R^* | T_P) \approx \frac{\frac{1}{m} \sum_{j=1}^m I(\pi_s^{(j)} \leq \pi_R^*) P(T_P | \boldsymbol{\pi}^{(j)})}{\frac{1}{m} \sum_{j=1}^m P(T_P | \boldsymbol{\pi}^{(j)})}. \quad [9]$$

It is important to emphasize that Eqs. [8] and [9] produce estimates for Eqs. [4] and [6]. Thus there is Monte Carlo error that should be taken into account. The amount of error will vary depending on the specific scenario and will be influenced by the number of components and their prior distributions among other factors. For more accurate estimates, the value m should be increased, but comes at the cost of increased computation time.

The equations in this section provide a framework for component assurance tests to augment system assurance tests, and were introduced in conjunction with a k-series system example. The rest of this section demonstrates how the approach can be generalized for practically any system that can be described using a reliability block diagram.

3.2. Almost-series systems

Consider a k-series system as shown in Figure 1. Previously we considered the case when all components fail independently, however, in practice that assumption is rarely true. If some dependency exists, it typically shortens the life of the system and the independence assumption produces an overly

optimistic estimate. Graves, Anderson-Cook, and Hamada (2010) refers to such a system as an almost-series system, (i.e., the system reliability is not exactly the product of the component reliabilities). In the more common case where the system reliability is less than the product of the component reliabilities, a latent component can be added (in series) to account for the discrepancy between the system, as constituted, and the system composed of independent components. This latent component can also represent an interface between the components whose reliability reduces the system reliability assumed by independent components.

We refer to this latent component as a “phantom” component. The prior on the phantom component is quite influential and must be selected with care. See Figure 2 for a reliability block diagram of a k-series system with a phantom component included.

When a phantom component is included in a model, it is basically modeled the same as any other component, with the exception that no data can be collected directly from it. We can quantify its effect through the prior denoted as $p(\pi_p)$. The only way the prior uncertainty on this phantom component can be reduced in the posterior is through full system tests. Thus, when phantom components are included in a system model, by necessity, some full system tests must be conducted and cannot be completely augmented by component tests to gain information about the phantom components.

As one might imagine, the outcome of the assurance test will be quite sensitive to the prior of this latent component. We suggest that, in most cases, the prior on π_p have most of its mass relatively near 1. One might interpret the meaning of π_p as the probability the system failed, and that failure could not be attributed to a single component. As the strength of the dependency between components increases, the need for a phantom component increases. Thus the mass of $p(\pi_p)$ should move further from 1.

To add a phantom component to a series system only a minor change is needed to the formulas presented in the previous section. We redefine π_s by replacing Eq. [1] with

$$\pi_s = f(\pi) = \pi_1 \times \cdots \times \pi_k \times \pi_p, \quad [10]$$

where π_p represents the probability the phantom component works and π now includes π_p . To find an



Figure 2. A k-component series system with an added phantom component.

assurance test for this system, again we compute Eqs. [8] and [9], but use $\pi_s^{(j)}$ as defined by Eq. [10].

3.3. Almost-parallel systems

We now briefly discuss a similar procedure for systems that are not completely parallel. But now, instead of a series of components, we use a series comprised of the parallel subsystem and a phantom component. See Figure 3 for a reliability block diagram of the system we are describing.

When adding a phantom component to an almost-parallel system, we change the definition of π_s to be:

$$\pi_s = f(\pi) = \pi_p \times [1 - (1 - \pi_1) \times \cdots \times (1 - \pi_k)].$$

Additionally, Eqs. [8] and [9] require a new definition for the probability that a test is passed, and a formula to obtain samples from the system prior distribution. They are:

$$P(T_P|\pi^{(j)}) = P(T_{P_s}|\pi_s^{(j)}) \left[1 - \prod_{i=1}^k P(T_{F_i}|\pi_i^{(j)}) \right],$$

and

$$\pi_s^{(j)} = \pi_p^{(j)} \left(1 - \prod_{i=1}^k (1 - \pi_i^{(j)}) \right).$$

Using this information we can develop an assurance test for an almost-parallel system.

3.4. Complex almost-series and almost-parallel systems

In this section, we present the most general version of our methodology by combining both almost-series and almost-parallel systems in a general framework. The system we consider is shown in Figure 4 and is essentially the same as found in Figure 3 of Guo, Jin, and Mettas (2011).

We will consider a modified version of this system in the application of Section 5. The reliability block

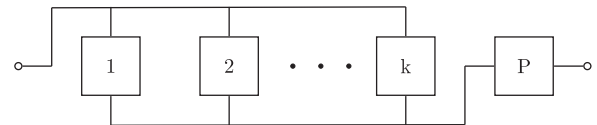


Figure 3. A k-component parallel system with a phantom component.

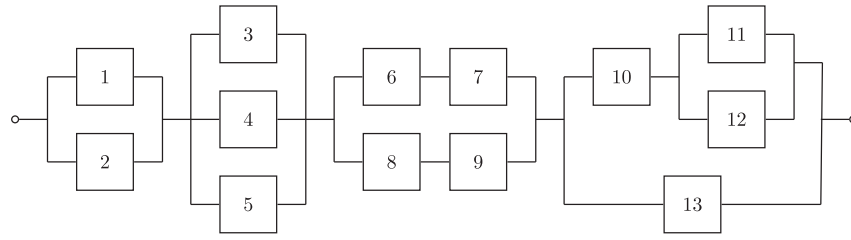


Figure 4. A complex system with components in series and parallel from Guo, Jin, and Mettas (2011).

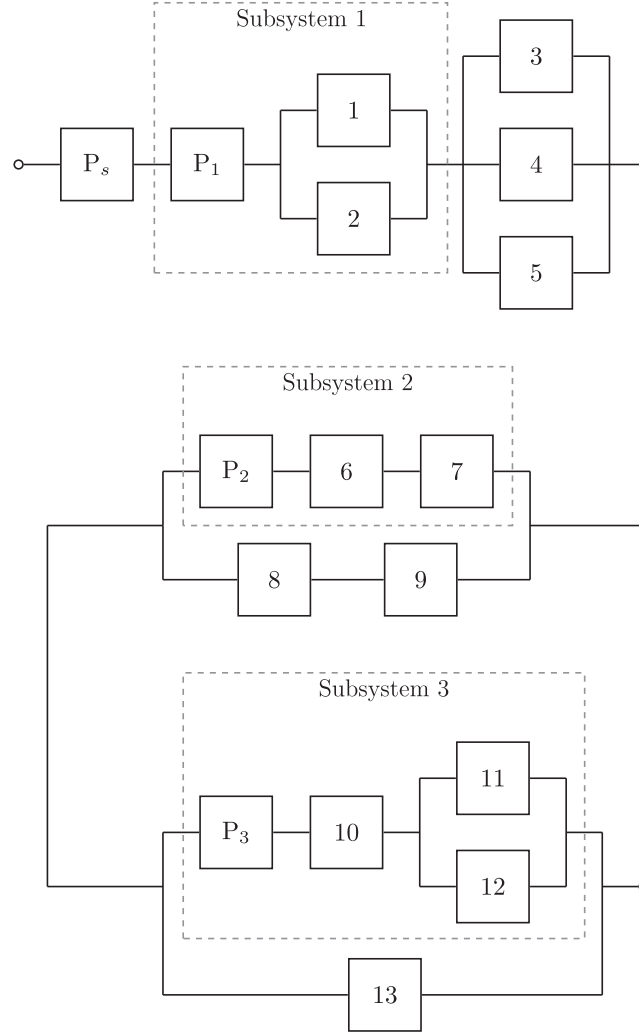


Figure 5. A modified version of the complex system from Figure 4 with three identified subsystems and four phantom components.

diagram of the modified version of this system is shown in Figure 5.

The formula to compute the system reliability is given in Eq. [11]. Additionally, the formula to compute the probability that the test is passed is shown in Eq. [12].

$$\begin{aligned} \pi_s^{(j)} = & \pi_{p_1}^{(j)} \left(1 - \prod_{i=1}^2 (1 - \pi_i^{(j)}) \right) \left(1 - \prod_{i=3}^5 (1 - \pi_i^{(j)}) \right) \times \\ & \left(1 - (1 - \pi_{p_2}^{(j)} \pi_6^{(j)} \pi_7^{(j)}) (1 - \pi_8^{(j)} \pi_9^{(j)}) \right) \times \\ & \left(1 - \left[1 - \pi_{p_3}^{(j)} \pi_{10}^{(j)} \left(1 - \prod_{i=11}^{12} (1 - \pi_i^{(j)}) \right) \right] (1 - \pi_{13}^{(j)}) \right) \pi_{p_s}^{(j)}. \end{aligned} \quad [11]$$

$$\begin{aligned}
P(T_P|\pi^{(j)}) &= P(T_{P_i}|\pi_s^{(j)})P(T_{P_{ss1}}|\pi_{ss1}^{(j)}) \\
&\times \left(1 - \prod_{i=1}^2 [1 - P(T_{P_i}|\pi_i^{(j)})]\right) \times \left(1 - \prod_{i=3}^5 [1 - P(T_{P_i}|\pi_i^{(j)})]\right) \\
&\times \left(1 - \left[1 - P(T_{P_{ss2}}|\pi_{ss2}^{(j)})\prod_{i=6}^7 P(T_{P_i}|\pi_i^{(j)})\right]\left[1 - \prod_{i=8}^9 P(T_{P_i}|\pi_i^{(j)})\right]\right) \\
&\times \left(1 - \left[1 - \prod_{i \in \{ss3, 10\}} P(T_{P_i}|\pi_i^{(j)})\right]\left(1 - \prod_{i=11}^{12} [1 - P(T_{P_i}|\pi_i^{(j)})]\right)\right) \\
&\times \left(1 - P(T_{P_{13}}|\pi_{13}^{(j)})\right).
\end{aligned} \tag{12}$$

Additionally,

$$\begin{aligned}
\pi_{ss1}^{(j)} &= \pi_{p_1}^{(j)} \left(1 - \prod_{i=1}^2 (1 - \pi_i^{(j)})\right), \pi_{ss2}^{(j)} = \pi_{p_2}^{(j)} \pi_6^{(j)} \pi_7^{(j)}, \text{ and} \\
\pi_{ss3}^{(j)} &= \pi_{p_3}^{(j)} \pi_{10}^{(j)} \left(1 - \prod_{i=11}^{12} (1 - \pi_i^{(j)})\right).
\end{aligned}$$

Although data cannot be obtained for phantom components directly, their effect can be updated with system or subsystem assurance tests in the posterior. We recommend a phantom component be added where all components may not fully describe the performance of the subsystem or system. However, adding too many phantom components could result in degraded performance or confusion about the role of each. For example in Figure 5, Subsystem 1 can be tested directly so that P_1 plays a useful role. However, if no data were collected on Subsystem 1, having two phantom components (P_s and P_1) at the system level would be redundant, and make it more challenging to determine the effect of their priors. Also in Figure 5 notice that a phantom component was not added in series with Components 8 and 9 since they do not comprise a subsystem that can be tested.

As mentioned earlier, this system will be used to demonstrate the extent of our methods and will be analyzed in Section 5.

Note that more complex forms of redundancy by a parallel structure can be implemented such as k-out-of-n systems or standby redundancy with an imperfect switch. More generally, the proposed method applies to any system whose reliability structure can be represented as a reliability block diagram.

3.5. Developing a component augmented assurance test

Thus far in the methodology, we have discussed the concept of augmenting system assurance with component

assurance tests, and provided formulas to compute the required quantities. In this section we show how all the pieces fit together and propose an algorithm to find a test plan that, in many cases, can reduce the overall cost of the test and still meets the consumer's and producer's risk thresholds.

Before we discuss the specific steps of our method, it is important to highlight a few items of import that are assumed and not discussed in much detail in this article. First, the test planner must have the system represented in a reliability block diagram (RBD). Any subsystems or components that can be tested directly should be identified. We denote l to be the number of lowest-level components in the system, where lowest-level components are items on the RBD that are not decomposed into smaller pieces. Phantom components would typically be considered as lowest-level components. We also refer to any subsystem, component, item, etc. that can also be tested as a "component" and use k to denote the number of those in the system. The relationship $l \leq k$ must hold. Next we assume that the cost of system and component tests are known. This can be somewhat subtle in that setting up a test for an item has associated costs, thus the first test usually absorbs those costs and the subsequent tests are less costly. Our approach can be applied to situations with variable costs (dependent on the number of tests), however in this article we assume fixed costs where the total cost of the test plan is the setup costs plus the cost for each test. In a true application of this method, there would be an initial setup cost (C_0). However, for the purpose of the illustrative examples below, we set $C_0 = 0$.

Prior distributions must be determined for the probability the component works for each of the l lowest-level components in the system. These parameters (denoted π_i) typically are given beta distribution priors. We note the somewhat counterintuitive behavior, that strong prior information on components does not necessarily lead to lower cost tests, especially when controlling PPR.

The k components are numbered and sorted. They should be sorted using the following criteria: sorted first by test expense (higher test cost items are listed before lower test cost items). In the event of ties, items in parallel should be placed before items in series (this could be a fairly subjective judgment in a complex system). In the event of any remaining ties, items whose probability of success has higher prior variance are placed before those with lower prior variance.

The primary goal of this article is to reduce the overall cost of an assurance test by augmenting system tests with component tests. We take a straightforward approach by first finding an assurance test that is composed solely of system tests. This is the baseline from which we start; and the goal is to improve (i.e., lower) the cost of this baseline test plan. Our algorithm is heuristic, and does not provide a guarantee to find the optimal test. However, we often find improvements over this established baseline, which is the current state-of-the-art approach.

The assurance test composed solely of system tests can be found using the algorithm defined in Figure 6, and is similar to Figure 10.1 found in Hamada et al. (2008). In Figure 6, \mathbf{n}_s and \mathbf{c}_s are vectors of length j_s and n_{s,j_s} or c_{s,j_s} refers to the last item in the respective vector. As another system test needs to be added, j_s is increased by a value of one and the empty variables n_{s,j_s} and c_{s,j_s} are set to $n_{s,j_s-1} + 1$ and c_{s,j_s-1} respectively. Thus j_s provides the current length of \mathbf{n}_s and \mathbf{c}_s . PPR and PCR also represent the respective posterior risks.

To implement our algorithm, we define a test score to establish preference of one test over another. If one test has $PPR \leq \alpha$ and $PCR \leq \beta$ and the other does not, the first test is preferred. Otherwise, we compute the following Assurance Test Score (ATS) or score for short for each test and compare, with the lower value being the preferable test. The score equation is:

$$ATS = \begin{cases} \sqrt{PPR^2 + PCR^2} & \text{if } PPR \leq \alpha \text{ and } PCR \leq \beta \\ PCR - \beta & \text{if } PPR \leq \alpha \text{ and } PCR > \beta \\ PPR - \alpha & \text{if } PPR > \alpha \text{ and } PCR \leq \beta \\ \sqrt{(PPR - \alpha)^2 + (PCR - \beta)^2} & \text{if } PPR > \alpha \text{ and } PCR > \beta \end{cases} \quad [13]$$

Our approach adds to the test found in Figure 6 by incorporating component tests. First, we start by finding a test using the algorithm in Figure 6. Using that test as a baseline, systems tests are then deleted one at a time. Between system deletions, component tests are added en masse until $PPR \leq \alpha$ and $PCR \leq \beta$. Then components tests are proposed for deletion one at a time. If $PPR \leq \alpha$ and $PCR \leq \beta$, the deletion is accepted. This is repeated sequentially for all components until a deletion no longer produces a valid test. The cost of the current test is then calculated and compared with the lowest costing (best) test thus far; if it is an improvement it is saved as the best test. If it is not the lowest costing test, then its cost is compared with the previously proposed test of the last iteration;

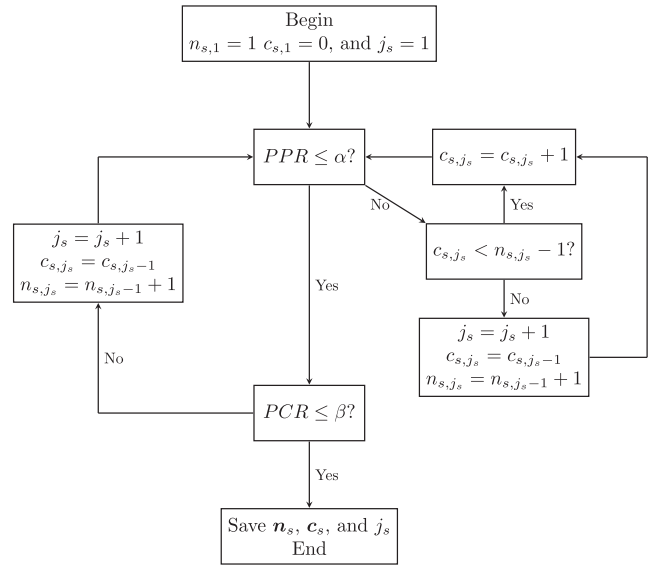


Figure 6. An algorithm to find a system assurance test that meets both the consumer's and producer's risk thresholds.

if it has lower cost than the previous test, we reset a counter. If an improvement has not been found in the 5 previous tests the algorithm is terminated. Otherwise, we continue to iterate the algorithm by deleting another system test. The number 5 is a tuning parameter and can be adjusted as computational resources allow. The full details for the algorithm described above are provided in Algorithm 1.

Algorithm 1: An algorithm to find a component augmented test

Input: $\mathbf{n}_s, \mathbf{c}_s$, and j_s (computed from algorithm in Figure 6), $m = 0$;
Step 1: Set $n_{i,1} = 0, c_{i,1} = 0$, and $j_i = 1$ for all i in $\{1, \dots, k\}$;
Step 2: $bestCost = testCost(\mathbf{n}_s, \mathbf{c}_s, j_s, \mathbf{n}_i, \mathbf{c}_i, j_i)$; $cost^p = bestCost$; $n_{i,b} = n_{i,j_i}$ and $c_{i,b} = c_{i,j_i}$ for all i in $\{s, 1, \dots, k\}$;
Step 3: $j_s = j_s - 1$; $r = 1$;
Step 4: if $j_s < 1$ then go to Step 26 end;
Step 5: $risks^c = test(\mathbf{n}_s, \mathbf{c}_s, j_s, \mathbf{n}_i, \mathbf{c}_i, j_i)$;
Step 6: if $(PPR \leq \alpha \text{ and } PCR \leq \beta)$ then go to Step 16 end;
Step 7: $score^c = ATS(risks^c)$;
Step 8: $(\mathbf{n}_i^+, \mathbf{c}_i^+, j^+) = addC(\mathbf{n}_i, \mathbf{c}_i, j_i)$;
Step 9: $(\mathbf{n}_i^-, \mathbf{c}_i^-, j^-) = subtractC(\mathbf{n}_i, \mathbf{c}_i, j_i)$;
Step 10: $risks^+ = test(\mathbf{n}_s, \mathbf{c}_s, j_s, \mathbf{n}_i^+, \mathbf{c}_i^+, j^+)$;
Step 11: $score^+ = ATS(risks^+)$;
Step 12: $risks^- = test(\mathbf{n}_s, \mathbf{c}_s, j_s, \mathbf{n}_i^-, \mathbf{c}_i^-, j^-)$;
Step 13: $score^- = ATS(risks^-)$;
if $(score^c \leq score^- \cap score^c \leq score^+)$ **then**
 $(\mathbf{n}_i, \mathbf{c}_i, j) = addN(\mathbf{n}_i, \mathbf{c}_i, j_i)$;
else if $(score^+ \leq score^-)$ **then**
 $\mathbf{n}_i = \mathbf{n}_i^+, \mathbf{c}_i = \mathbf{c}_i^+$, and $j = j^+$;

```

else
   $n_i = n_i^-, c_i = c_i^-$ , and  $j = j^-$ ;
end
Step 14:  $r = r + 1$ ;
Step 15: if ( $r < 10$ ) then go to Step 5 else go to
Step 3 end;
Step 16:  $i = 1$ ;
Step 17: Go to next statement;
if  $n_{i,j_i} = 0$  then
   $i = i + 1$ ;
  if  $i > k$  then
    go to Step 22
  else
    go to Step 17
  end
Step 18:  $j_i = j_i - 1$ ;
Step 19: Compute test( $n_s, c_s, j_s, n_i, c_i, j$ );
Step 20: if  $PPR > \alpha$  or  $PCR > \beta$  then  $j_i = j_i + 1$ ;
 $i = i + 1$  end;
Step 21: Go to Step 17;
Step 22:  $cost^c = testCost(n_s, c_s, j_s, n_i, c_i, j)$ ;
  if  $cost^c < cost^p$  then
     $m = 0$ ;
  else
     $m = m + 1$ 
  end
Step 23:  $cost^p = cost^c$ ;
Step 24: if ( $cost^c < bestCost$ ) then  $bestCost = cost^c$ ;
 $n_{i,b} = n_{i,j_i}$  and  $c_{i,b} = c_{i,j_i}$  for all  $i$  in  $\{s, 1, \dots, k\}$  end;
Step 25: if ( $m < 5$ ) then go to Step 3 end;
Step 26: Stop;
Output:  $n_{i,b}$  and  $c_{i,b}$  for all  $i$  in  $\{s, 1, \dots, k\}$ 

```

We include several items of clarification on the notation for Algorithm 1. First, for each component i , n_i, c_i are vectors of length j_i , with elements $n_{i,a}, c_{i,a}$ respectively, for $a = 1, \dots, j_i$. Additionally, j contains the values j_i for $i \in \{1, \dots, k\}$, and k is the number of components. Next, the function *test* takes the values n_{i,j_i} and c_{i,j_i} for all $i \in \{s, 1, \dots, k\}$, computes the *PPR* and *PCR*, and returns these two values in a vector. The assurance test score function *ATS* uses the inputs of the *PPR* and *PCR* and applies Eq. [13]. The function *testCost* takes the values n_{i,j_i} for all i in $\{s, 1, \dots, k\}$ and returns the cost of that number of system and component tests. The function *addC* adds a value of one to each j_i and then sets $c_{i,j_i} = c_{i,j_i-1} + 1$ for all i in $\{1, \dots, k\}$, if any of the resulting $c_{i,j_i} \geq n_{i,j_i-1}$ then $n_{i,j_i} = c_{i,j_i-1} + 1$ otherwise $n_{i,j_i} = n_{i,j_i-1}$. For all c_{i,j_i} that are greater than 0, the function *subtractC* sets $c_{i,j_i} = c_{i,j_i} - 1$. The function *addN* adds a value of one to each j_i and then sets $n_{i,j_i} = n_{i,j_i-1} + 1$ and $c_{i,j_i} = c_{i,j_i-1}$ for all i in $\{1, \dots, k\}$.

The variable r was introduced to ensure the algorithm did not get stuck in a infinite loop. The higher the value r the longer the algorithm might take. In Step 15, the value 10 is also a tuning parameter and its value should be somewhat dependent on the prior for the system phantom component. If the prior has significant mass away from 1, a higher value should be considered.

In this section, we have provided the details of our proposed method. The following section provides some insights into how this method can improve costs using a few straightforward simulation scenarios.

4. Simulation study

In this section, we conduct a simulation study to show how component tests can augment assurance tests. The goal of this simulation is to show that under certain circumstances component assurance tests can be used to replace system assurance tests and produce an overall lower cost for the assurance test.

In this simulation we consider four simple systems that are key building blocks to a reliability block diagram, as shown in Figure 7. The systems are a 2-component series system, a 2-component parallel system and each of these with the addition of a phantom component.

The simulations were run under the following conditions. First, each component's reliability parameter π_i had a prior distribution of Beta(8, 4). Next, the phantom component's reliability parameter π_p was given a prior of Beta(11, 1). Additionally, the following values were used for the test conditions, $\alpha = \beta = 0.05$, $\pi_A^* = 0.95$, and $\pi_R^* = 0.90$. Finally, we used generic cost units and set the test costs for each component to be 1 unit and a system test to cost 4 units, illustrative of the situation where system tests are usually more expensive than component tests.

The algorithm found in Figure 6 and Algorithm 1 were run for each system displayed in Figure 7. The simulation results are found in Figure 8 and Table 1. The simulation shows that indeed the costs of the tests go down as component tests replace system tests. Additionally, for three of these cases, the parallel structure and/or the phantom components do not allow the number of system tests to go to zero. However, for the series system with no phantom component, the system tests can be completely replaced by component tests (using the specific costs in this scenario).

In each of the four systems, we obtained a lower cost test than using only system tests. When phantom components were used, we can see our algorithm stop when a lower cost test cannot be found and well before all system tests are replaced. For the parallel

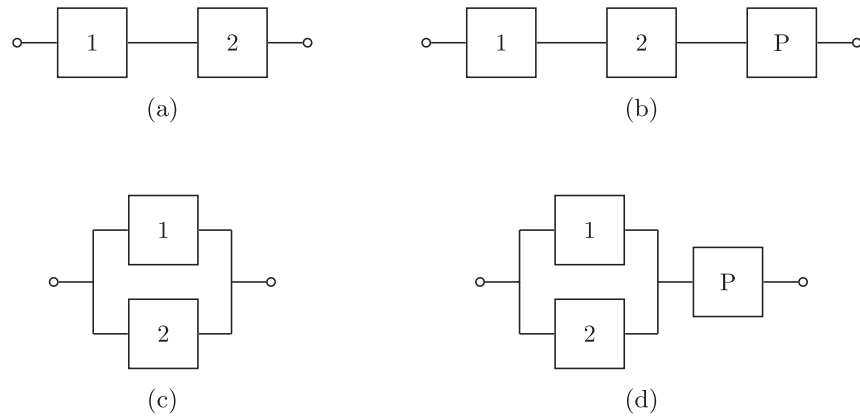


Figure 7. The four systems that are incorporated into the the simulation study.

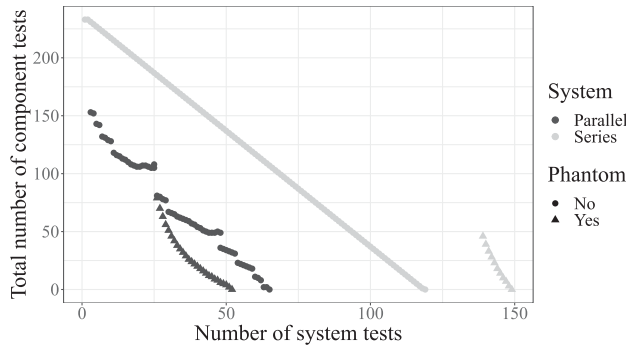


Figure 8. A plot of the simulations as component tests replace system tests. These are based on the four systems displayed in Figure 7.

Table 1. Simulation results based on system and component tests for the four systems displayed in Figure 7.

System Figure		7(a)	7(b)	7(c)	7(d)
System Tests Only	n_s	119	149	65	52
	c_s	0	0	3	0
	Cost	476	596	260	208
	Time	61s	116s	54s	47s
System & Component Tests	n_s	0	146	4	33
	c_s	0	0	1	0
	$\sum n_i$	237	10	144	38
	$\sum c_i$	0	0	23	0
	Cost	237	594	160	170
	Time	737s	217s	954s	355s
Cost Difference		-239	-2	-100	-38
Time Difference		676s	101s	900s	308s

system without a phantom component (as shown in Figure 7c), we see dramatic jumps in the number of component tests needed in the process. This occurs when c_s decreases by a value of 1. This necessitates increasing the number of acceptable component failures in order to control PPR which can lead to an unacceptable PCR and then requires more component tests. These jumps do not occur when not controlling for PPR or when $c_s = 0$. The computation times are also shown in Table 1. Clearly, the computational penalty for finding a cheaper test is light when

compared with test costs. In these simulations we used a Monte Carlo sample size of 1,000,000. For our computation in this article we used a computer with 92 GB of RAM and a Intel Xeon Gold 6142 CPU that runs at 2.60 GHz.

This simulation shows that it can be quite advantageous to augment system tests with component tests in an assurance test.

5. Application

In this section, we consider a system similar to that presented by Guo, Jin, and Mettas (2011) and later in Hamada et al. (2014). We implement our method on the system given in Figure 5 in two specific scenarios. First, we compare with the results in Hamada et al. (2014) without controlling for PPR. For the second scenario, we control for PPR, and use Jeffreys reference priors on the components.

In the first scenario, the Monte Carlo sample sizes used to compute the formulas were set at 100,000. The parameter values used in both scenarios were $\beta = 0.05$ and $\pi_R^* = 0.95$. When comparing to the results from Hamada et al. (2014) we set $\alpha = 1$ and do not consider π_A^* , however, when controlling for PPR, we set $\pi_A^* = 0.96$ and $\alpha = 0.05$. We developed notional costs in arbitrary units for the system and each component test as shown in Table 2. In Tables 2, S1, S2, and S3 represent the three subsystems and S represents the whole system.

Hamada et al. (2014) considers the system in Figure 4 and found that 7 system tests were required for PCR to be less than 0.05. They used informative Beta(a , b) priors for each component. The values of a for each component are listed in Table 3 with $b = 1$ for all components (we use the same component priors in both examples).

We use our method on the system in Figure 5, which includes the extra four phantom components.

Table 2. Costs for system and component tests.

Component	1	2	3	4	5	6	7	8	9
Cost of 1 Test	10	10	9	9	9	6	6	6	6
Component	10	11	12	13	S1	S2	S3		
Cost of 1 Test	8	5	5	19	25	15	21	200	

Table 3. Component priors, results, and costs when comparing with Hamada et al. (2014).

Component	1	2	3	4	5	6	7	8	9	10	11
Value of a	26	26	21	21	12	12	12	16	16	16	16
n_i	0	0	0	0	0	0	9	1	12	4	0
Test Cost	0	0	0	0	0	36	6	72	24	0	0
Component	12	13	S1	P1	S2	P2	S3	P3	S	PS	Σ
Value of a	21	21	–	500	–	500	–	500	–	500	–
n_i	0	0	0	–	0	–	0	–	0	–	26
Test Cost	0	0	0	–	0	–	0	–	0	–	156

We used Beta(500, 1) as the priors for the phantom components. After running the algorithm in Figure 6 we find that, if $n_s = 13$ and $c_s = 0$, then $PCR \leq \beta$. The value of n_s using our method is higher, by 6 system tests, because of the incorporation of the phantom components. After applying Algorithm 1, we find the number of system and component tests needed for $PCR \leq \beta$. Those values are displayed in Table 3. The computational costs were 3 s for the system only test and 79 s for the component augmented test.

The cost for the test developed by Hamada et al. (2014) is 1,400 units. When incorporating component tests using our method, we find the total cost is 156 units, which is a huge reduction in cost. Thus, we obtain a less expensive test and allow for some slight deviations from the independence assumptions by augmenting with component tests. Note that in this example we are more cost effective because the components are cheaper to test in this scenario. However, if the costs of the components were higher, relative to the subsystem and system tests, the benefit would be reduced.

In the second scenario, we still consider the system described in Figure 5, but now control for PPR, and compare our component augmented test with the method that uses only system tests. To demonstrate a variety of situations, in this scenario we use Jeffreys reference priors, Beta (0.5, 0.5), for all lowest-level components (of which there are 13). For the phantom components, we use the previous priors of Beta(500, 1).

Using the algorithm in Figure 6, a test plan with $n_s = 108$ and $c_s = 0$ satisfied the risk requirements. This test costs 21,600 units. When augmenting with component tests, we were able to eliminate all system tests and the specific component tests are shown in Table 4, with a cost of 7,902 units. Again the reduction in cost is dramatic, but these results are highly dependent on the test costs of the components relative

Table 4. Component priors, results, and costs when controlling for PPR.

Component	1	2	3	4	5	6	7	8	9	10	11
Value of a and b	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
n_i	74	66	53	55	84	100	101	107	108	81	10
Test Cost	0	0	0	0	0	0	0	0	0	0	0
Component	12	13	S1	P1	S2	P2	S3	P3	S	PS	Σ
Value of a and b	0.5	0.5	–	500	–	500	–	500	–	500	–
n_i	9	65	12	–	0	–	0	–	0	–	925
Test Cost	0	0	0	–	0	–	0	–	0	–	7902

to those of the system. The computational costs were 18 s for the system only test and 1,137 s for the component augmented test.

This application has shown that using component tests in a disciplined way can decrease test costs while also allowing for some deviations in the independence assumptions.

6. Discussion

In this article, we presented a novel method to use component tests to augment system tests in an assurance test. In many situations this augmentation can provide an overall lower cost assurance test that satisfies the producer's and consumer's risk thresholds. We demonstrated in simulations and an application significant cost improvements which could be widely utilized in assurance testing with constrained budgets.

One referee asked whether cost in terms of testing time and a financial budget could be used. The costs in our method are generic so they could be testing times. If the assurance test found by our method does not meet a budget, the consumer and producer would need to negotiate increasing the risks' thresholds and/or the budget.

For choosing prior distributions for component reliabilities, in addition to available industry knowledge, the producer may have relevant component test data. Nevertheless, the producer and consumer need to agree on them before designing the assurance test.

Future work might seek improvements over the heuristic algorithm offered here for replacing system assurance tests with component tests to optimize cost. Additionally, we only consider one-shot systems, but system reliability can be measured in other ways. Therefore, future work could extend these findings to systems with other failure types.

About the authors

Richard L. Warr is an Associate Professor of Statistics at Brigham Young University. Dr. Warr graduated Magna Cum Laude with a BS in Mathematics from Southern Utah

University in 1996. He earned an MA in Mathematics from the University of Nebraska at Omaha in 2005 and his MS and PhD degrees in Statistics at the University of New Mexico in 2009 and 2010. Dr. Warr's research is focused on statistical methodology, specifically in the areas of reliability, random partition models, computational methods, and stochastic processes. In these areas he often uses a Bayesian nonparametric approach.

Jace E. Ritchie is currently a PhD student at Utah State University. Jace graduated Summa Cum Laude with a BS and MS in Statistics from Brigham Young University in 2023 through an integrated program. Jace's research has included optimal design of experiments, evolutionary algorithms, reliability, Bayesian statistics, and automobile crash safety. He also has experience utilizing machine learning to predict E-commerce metrics.

Michael Hamada is a retired Scientist from Los Alamos National Laboratory (LANL). He earned a Ph.D. in Statistics from the University of Wisconsin-Madison. He is a Fellow of LANL, ASQ, and ASA. His research interests include design of experiments, reliability, quality control, and measurement system assessment.




Acknowledgements

Data sharing is not applicable to this article as no new data were created or analyzed in this study. We thank two anonymous referees whose insightful comments on an earlier version helped significantly improve this article.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Richard L. Warr  <http://orcid.org/0000-0001-8508-3105>
Jace Ritchie  <http://orcid.org/0009-0008-1109-3974>
Michael Hamada  <http://orcid.org/0000-0003-3206-1695>

References

- Graves, T. L., C. M. Anderson-Cook, and M. S. Hamada. 2010. Reliability models for almost-series and almost-parallel systems. *Technometrics* 52 (2):160–71. doi: [10.1198/TECH.2009.08185](https://doi.org/10.1198/TECH.2009.08185).
- Guo, H., T. Jin, and A. Mettas. 2011. Designing reliability demonstration tests for one-shot systems under zero component failures. *IEEE Transactions on Reliability* 60 (1):286–94. doi: [10.1109/TR.2010.2085552](https://doi.org/10.1109/TR.2010.2085552).
- Hamada, M. S., A. Wilson, C. S. Reese, and H. Martz. 2008. *Bayesian reliability*. New York, NY: Springer.
- Hamada, M. S., A. G. Wilson, B. P. Weaver, R. W. Griffiths, and H. F. Martz. 2014. Bayesian binomial assurance tests for system reliability using component data. *Journal of Quality Technology* 46 (1):24–32. doi: [10.1080/00224065.2014.11917952](https://doi.org/10.1080/00224065.2014.11917952).
- Li, M., W. Zhang, Q. Hu, H. Guo, and J. Liu. 2017. Design and risk evaluation of reliability demonstration test for hierarchical systems with multilevel information aggregation. *IEEE Transactions on Reliability* 66 (1):135–47. doi: [10.1109/TR.2016.2619689](https://doi.org/10.1109/TR.2016.2619689).
- Meeker, W. Q., and L. A. Escoba. 2004. Reliability: The other dimension of quality. *Quality Technology & Quantitative Management* 1 (1):1–25. doi: [10.1080/16843703.2004.11673062](https://doi.org/10.1080/16843703.2004.11673062).
- Robert, C. P., and G. Casella. 2004. *Monte Carlo statistical methods*. Berlin, Heidelberg: Springer-Verlag.
- Smith, C., D. Kelly, and H. Dezfuli. 2010. Probability-informed testing for reliability assurance through Bayesian hypothesis methods. *Reliability Engineering & System Safety* 95 (4):361–8. doi: [10.1016/j.res.2009.11.006](https://doi.org/10.1016/j.res.2009.11.006).
- Ten, L. M., and M. Xie. 1998. Bayes reliability demonstration test plan for series-systems with binomial subsystem data. In *Annual Reliability and Maintainability Symposium. 1998 Proceedings. International Symposium on Product Quality and Integrity*, 241–6. IEEE.
- Wilson, K. J., and M. Farrow. 2021. Assurance for sample size determination in reliability demonstration testing. *Technometrics* 63 (4):523–35. doi: [10.1080/00401706.2020.1867646](https://doi.org/10.1080/00401706.2020.1867646).
- Wilson, A. G., and K. M. Fronczyk. 2017. Bayesian reliability: Combining information. *Quality Engineering* 29 (1): 119–29.