Reliability Demonstration for Safety-Critical Systems

Oded Tal

RAFAEL, Haifa, Israel

Chris McCollin*

Department of Mathematics, Statistics and Operational Research, The Nottingham Trent University, UK

* Corresponding author. Fax: +44 115-8482998

Tony Bendell

The Management Centre, University of Leicester, UK

Key Words- hardware and software reliability, reliability demonstration, safety-critical systems, statistical testing, sequential testing.

Summary

This paper suggests a new model for reliability demonstration of safety-critical systems, based on the TRW Software Reliability Theory. The paper describes the model; the test equipment required and test strategies based on the various constraints occurring during software development. The paper also compares a new testing method, Single Risk Sequential Testing (SRST), with the standard Probability Ratio Sequential Testing method (PRST), and concludes that:

- **SRST provides higher chances of success than PRST.**
- **SRST takes less time to complete than PRST.**
- **SRST satisfies the consumer risk criterion, whereas PRST provides a much smaller consumer risk than the requirement.**

Notation

F	number of failures during reliability demonstration testing
F_{max}	maximum allowable number of failures during reliability demonstration testing
n	number of tests
n_0	the number of tests required for zero-failure demonstration
n_F	the number of tests required for reliability demonstration with F failures
n_{max}	the maximum number of tests which can be performed in T_{max} testing hours.
T_{max}	the total time available for testing

t the time left until the deadline [hours]

T_0 the required time for zero-failure demonstration [hours]

T_{ave} the average duration of a single test

T_{corr} the average time needed in order to find the fault(s), correct them, do the regression tests on the new version and return it to statistical testing [hours]

T_F the time required for reliability demonstration with F failures [hours]

V the number of the current software version

V_{max} the maximum number of new software versions that can still be produced by the deadline

c the required confidence that the probability of a failure during one mission/demand is less than θ_1

α producer risk

β consumer risk

θ_1 the maximum allowed unreliability corresponding to the customer risk

θ_0 specified unreliability corresponding to the producer risk

θ the true unreliability of the system

1. INTRODUCTION

At present there are two generally accepted methods for reliability demonstration testing, i.e. fixed-duration testing and sequential testing, as specified by US MIL-HDBK-781A [1]. Sequential testing, also known as Probability Ratio Sequential Testing (PRST), is more efficient in terms of the mean number of tests. Both methods are based on the concepts of consumer risk β and producer risk α and on two corresponding unreliability values (a specified failure rate and a minimum acceptable failure rate) and tests can be chosen from the appropriate operating characteristic curve within US MIL-HDBK 781A based on these four criteria. This approach is appropriate for non-safety-critical systems, but it is less appropriate for safety-critical systems, because:

1. PRSTs within [1] are based on specified and minimum acceptable failure rates. PRSTs for software reliability testing may be based on acceptance sampling by attributes where the success or failure of a program is more relevant than its time to failure. The background theory is described in [2], [3] and [4]. A complete review of software reliability models appears in [5].
2. In the context of safety-critical systems, the producer risk and the specified failure rate or unreliability are not customer requirements. Nevertheless, they do affect the required number of tests and the maximum allowed number of failures during the reliability demonstration testing.

3. While for non-safety-critical systems the producer risk and the consumer risk can be interpreted in comparable financial terms, i.e. a financial loss is experienced by consumer and producer if the test fails, for safety-critical they usually are not comparable, since a failure in such a system can result in extensive non-reversible damage in terms of human life and/or environmental effects.
4. The maximum number of tests for safety-critical systems can be very large for a PRST since the testing ends according to two truncation criteria, i.e. a maximum number of tests or a maximum number of failures which depend on the four test criteria. The maximum number of failures can be very large too, which is undesirable for safety-critical systems.

2. BACKGROUND

A contract for software reliability/dependability for a safety critical software system specifies the following requirements: minimum acceptable unreliability θ_1 with a corresponding consumer risk β and a delivery date. The test time up to the delivery date, T_{max} hours, can be extremely long (upwards of 8,000 hours) to meet a reliability requirement so a specific model and test equipment have been developed [6] and are described under sections 3, 4 and 5 of this paper. Test strategies involving concerning T_{max} , T_{ave} and the best times to run a demonstration for military avionics are presented in sections 6 and 7. In section 8, an optimal testing strategy is considered.

The model for software reliability demonstration for safety-critical systems is based on the Balls and Urn model of software reliability [7], the TRW software reliability theory [8], the SRST method for software reliability demonstration and the Test-Analyse-And-Fix (TAAF) concept [9].

3. MODEL ELEMENTS: THE BALLS AND URN MODEL

The Balls and Urn model for software reliability includes two components. The urn represents a software program and balls represent all the possible input values to the software; white balls represent inputs which cause the software to function properly and black balls represent inputs which cause the software to fail. According to TRW Software Reliability Theory, the software reliability of a computer program is the probability that the software will not fail on some input case from a given input distribution. Applying this definition to the Balls and Urn model, the software reliability of a program becomes the proportion of white balls in the urn and conversely, the software unreliability becomes the proportion of black balls. Therefore it is possible to estimate the software reliability by randomly selecting balls from the urn, recording their colors and returning them to the urn. If the program is executed a large number of times with random inputs from this distribution (the operational profile of the

software), the observed reliability will converge to the operational reliability of the software. The model for reliability demonstration of safety critical systems by statistical testing is based on the Balls and Urn Model with some changes and additions:

1. The reliability of a system is the probability that it will not fail during one mission or demand, according to the system's type.
2. Each ball represents a trajectory of input values to the software, or a whole mission/demand. Therefore the software is tested with randomly selected mission cycles/demands.
3. The random selection applies not only to input data values, but also to their sequence and timing within each cycle/demand, as well as to the total length of each mission cycle demand.
4. The black balls represent mission profiles/demands which lead to failures. Upon finding such a failure that mission cycle/demand is stopped, and another started.
5. The purpose of the reliability demonstration is to demonstrate a certain reliability with a certain confidence level, with or without any failures. Therefore, it is a process of one or more testing cycles, and not a "one-shot" test. It may or may not have a time limit.

The purpose of the model is to find the optimal testing policy for reliability demonstration testing, according to minimum testing time, maximum probability of success or minimum cost. The same model can be used for dependability demonstration, provided only catastrophic failures are taken into account.

Figure 1 is a schematic description of the Balls and Urn model elements for an avionics system. The relevant features of the system and its development environment are the mission/demand duration distribution, the operational profile of the system, the time required to fix the software and produce a new version, including regression testing and the estimated failure rate of the software.

The Mission Profile Generator simulates the system's inputs, and its outputs are fed to the Oracle and recorded by the Monitor where:

- The Mission Profile Generator is a real-time software system, which use the operational profile, the mission/demand duration distribution and a pseudo-random generator to produce random mission profiles/demands. These mission profiles are fed to the avionics system and to the Oracle, and recorded by the Monitor. The generated mission profiles have random length and random inputs, and these inputs are also randomly timed and sequenced.

- The Oracle is a real-time system, which examines the inputs and the outputs of the safety-critical system, and perfectly detects any failures. Its function depends, of course, on a

prior definition of all possible failures. The Oracle receives the system's inputs from the MPG, and the system's outputs directly from the system itself.

- The Monitor is a real-time software system, which continuously records all the inputs and the outputs of the system, in order to enable debugging in case of finding a failure. The monitor receives the system's inputs from the MPG, and the system's outputs directly from the system itself.

4. MODEL ELEMENTS: THE TRW MODEL AND THE SRST

4.1 The TRW Model

The assumptions of the TRW Model are:

1. θ is unknown.
2. All the tests are s-independent, and represent full mission cycles of the system.
3. The contract does not specify α nor the corresponding θ_0.

It is derived as follows. The probability of success of a single test is $1-\theta$. Therefore the probability of success of n independent tests is $(1-\theta)^n$**,** and the probability of at least one failure in n tests is $1-(1-\theta)^n$**.** If no failures are found during n tests then the probability that θ is larger than any required θ_1 is

$\beta=(1-\theta_1)^n$ (reference [8])

For any given β and n, θ_1 is the $(1-\beta)$ upper confidence bound on θ. The required number of tests with no failures for any given values of θ_1 and β is the integer solution of:

$n=\ln(\beta)/\ln(1-\theta_1)$

The same kind of reasoning can be applied to any number of failures, F:

$$\sum_{j=0}^{F} nCj.\,(1-\theta_1)^{n-j}\,.\theta_1^{\,j} \le \beta \qquad j=0,1,2....F \text{ and } nCj = n!/(j!(n-j)!) \qquad (1)$$

The required number of tests for any given β,θ_1 and F is the smallest n which satisfies *(1)*. Numerical solutions for equation *(1)* have been calculated and are tabulated in Table 1. Three alternative empirical formulae also have been used for comparison with Table 1 and are discussed in [6]. Each method gives similar results.

Table 1: The required number of tests for various values of θ_1 and c and F=0-10

Note: only n<100,000 have been included.

c=0.90

θ_1/**F**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.1	22	38	52	65	78	91	104	116	128	140	152
0.05	45	77	105	132	158	184	209	234	258	282	306
0.01	230	388	531	667	798	926	1051	1175	1297	1418	1538
0.005	460	777	1063	1335	1597	1853	2105	2352	2597	2839	3079
0.001	2302	3889	5321	6679	7992	9273	10530	11769	12993	14204	15404
0.0001	23025	38896	53222	66806	79934	92745					

c=0.95

θ_1/**F**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.1	29	46	61	76	89	103	116	129	142	154	167
0.05	59	93	124	153	181	208	234	260	286	311	336
0.01	299	473	628	773	913	1049	1182	1312	1441	1568	1693
0.005	598	947	1258	1549	1829	2100	2362	2627	2884	3138	3389
0.001	2995	4742	6294	7752	9151	10511	11840	13146	14432	15702	16959
0.0001	29956	47437	62956	77535	91533						

c=0.99

θ_1/**F**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.1	44	64	81	97	113	127	142	156	170	183	197
0.05	90	130	165	198	229	259	288	316	344	371	398
0.01	459	662	838	1001	1157	1307	1453	1596	1736	1874	2010
0.005	919	1325	1678	2006	2318	2618	2910	3196	3476	3752	4024
0.001	4603	6636	8403	10042	11601	13105	14567	15996	17398	18779	20140
0.0001	46050	66381	84057								

c=0.995

θ_1/**F**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.1	51	72	90	106	122	137	152	167	181	195	209
0.05	104	146	182	216	248	279	309	338	367	395	423
0.01	528	740	924	1094	1256	1411	1562	1709	1853	1995	2135
0.005	1058	1483	1852	2192	2515	2826	3128	3422	3711	3995	4274
0.001	5296	7427	9271	10974	12590	14146	15655	17129	18573	19993	21393
0.0001	52981	74299	92735								

c=0.999

θ_1/**F**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0.1	66	89	108	126	143	159	175	190	205	220	235
0.05	135	181	220	257	291	324	356	387	417	447	476
0.01	688	920	1119	1302	1475	1640	1801	1957	2110	2259	2407
0.005	1379	1843	2242	2608	2954	3286	3607	3919	4225	4525	4820
0.001	6905	9230	11225	13058	14789	16450	18056	19620	21150	22651	24127
0.0001	69075	92331									

4.2 Determining the SRST

SRST is the following steps:

1. Use T_{max} and T_{ave} to calculate n_{max}.
2. Find all the possible pairs (n,F) satisfying the above inequality, subject to F=0,1,2,... and $n \le n_{max}$. The largest F is F_{max} during n_{max} tests.
3. Plot the graph of F=f(n) using the computed (n,F) pairs, the horizontal line $F=F_{max}$ and the vertical line $n= n_{max}$. These graphs divide the n-F plane into three areas:
5. $F>F_{max}$: reject the software.
6. $F\le F_{max}$ and to the left of F=f(n): continue testing.
7. $F\le F_{max}$ and to the right of F=f(n): accept the software.

Example: For $\beta=0.05$, $\theta_1=0.001$, $T_{max}=12{,}000$ hours, $T_{ave}=1$ hour, $n_{max} =T_{max}/T_{ave}=12{,}000$

Table 2 gives the relevant (n,F) pairs, found by numerical methods.

Table 2: (n,F) pairs

F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
N	2995	4742	6294	7752	9151	10511	11840	13146

Therefore $F_{max}=6$ and the precise n_{max} is 11,840. Figure 2 is the graphical decision aid. The required reliability can be demonstrated by any of the above (n,F) pairs, e.g. 2995 tests with 0 failures, or 4742 tests with 1 failure, etc.

The probability of accepting the product is the probability of finding F_{max} or less failures in n tests, where (F_{max},n) is any integer point on the acceptance line. Therefore it can be calculated by:

$$\beta(\theta,n)= \sum_{j=0}^{F\,\max} nCj\ (1-\theta)^n\theta^j$$

and the power of the test is:

$$\text{power}(\theta,n)=1-\beta(\theta)=1- \sum_{j=0}^{F\,\max} nCj\ (1-\theta)^n\theta^j$$

For the SRST, since the acceptance line was calculated according to the cumulative binomial probability formula, for a given θ, the power of the test does not change along the acceptance line. For a given pair (F_{max},n), the power of the test increases very rapidly with θ. For the specified value of θ_1, the calculated consumer's risk, $\beta(\theta_1)$, is identical to the required value, 0.05.

4.3 A comparison of PRST and SRST

Statistical testing is based on the concept of hypothesis testing. In the simplest case the null hypothesis and the alternative hypothesis are as follows:

H_0: $\theta\le\theta_1$

H_1: $\theta>\theta_1$

There are two types of errors associated with hypothesis testing [10]:

1. Type I error: H_0 is true, but it is rejected following the test. The probability of this error is denoted by α and is known as the level of significance of the test.
2. Type II error: H_0 is false, but it is accepted following the test. The probability of this error is denoted by β, and $1-\beta$ is known as the power of the test.

Both α and β are functions of the real, unknown θ, and should be minimised. For a given sample size (n), a decrease in one type of error leads to an increase in the other type. In fact, for a simple one-tailed test, $\beta(\theta)=1-\alpha(\theta)$, and the only way to simultaneously decrease both types of error is to increase the sample size.

The calculations below are based on the following data: $\theta_1=0.001$, $\theta_0=0.0005$, $\alpha=\beta=0.05$, for which the PRST graphical aid is depicted in Figure 3.

For a given θ, the power of the test changes only slightly along the acceptance line for different pairs of (F_{max}, n). For a given pair F, the power of the test increases very rapidly with θ. For the specified value of θ_1, the calculated consumer risk $\beta(\theta_1)$ is 0.00276 which is much better than the required value, 0.05. This applies to any combination of α, β, θ_0, θ_1 and has been reported in previous works such as [11], [12], [13], [14].

SRST can be regarded as a combination of sequential testing and fixed-duration testing, since the required reliability can be demonstrated at the end of one of several fixed-duration periods, corresponding to different numbers of failures. SRST is better than PRST because of the following reasons:

1. The law of parsimony: SRST requires only two parameters, β and θ_1, instead of four in PRST.
2. Efficiency: For reasonable values of α and θ_0 in the PRST method, the required number of tests for any given number of allowable failures is smaller according to the SRST method. Therefore, for software which is good enough, SRST requires less time and costs less.
3. Probability of success: The use of F_{max} as the rejection criterion, instead of a second line, parallel to the acceptance criterion line, makes SRST more liberal than PRST in cases when the failure density is larger at the beginning of the tests and decreases later. Therefore the probability of success is larger in SRST.
4. Truncation criteria: in PRST the "truncation criteria", F_{max}, and n_{max}, are derived from real-life constraints, i.e. a deadline or the time allocated to the reliability demonstration, and therefore cannot be very large, as is often the case with PRST.
5. The SRST method is simpler to understand and to implement than the PRST method, and it does not require a graphical decision rule.

6. Although the power of the SRST method is always smaller, it does satisfy the required consumer's risk criterion.

For the PRST, All in all, using SRST for reliability demonstration of safety-critical systems is much better than using the standard PRST method, both from the producer's and the consumer's point of view. The PRST has attributes not available to the SRST; namely the Operating Characteristic (OC) function and the average sample number (ASN) used as aids to define appropriate tests. However, for SRST, the two parameters, β and θ_1 are defined early on in a contract allowing enough time to determine the appropriate SPRT from table 1.

4.4 Duration and probability of success

In order to demonstrate the superiority of the SRST method in terms of its expected duration and probability of success a simulation was used. The simulation was based on the assumption that the time between failures is exponentially distributed. Three values of real unreliability were used: 0.0005, 0.001 and 0.002. For each of these values 100 different failure-trajectories were simulated, and then examined according to the PRST and SRST decision rules. In order to compare the two methods a truncated PRST method had to be designed. For a given number of tests as one of the truncation criteria, n_{max}, the second truncation criterion, F_{max}, can be calculated from the following formula [12]:

$n_{max}=(1/2\theta_0)\chi^2_{(1-\alpha),2Fmax}$

For n_{max}=6300 tests, the maximum allowable number of failures according to the SRST method is 2. According to the PRST method, the maximum number of failures on the rejection line is 7. The results of the simulation are listed in Table 2:

Table 2: The probability of success and the expected number of tests for both methods

Method	θ	PRST	SRST
Probability of success	0.0005	0.08	0.47
Mean number of tests to accept		5889 tests	4160 tests
Mean number of tests to reject		6230 tests	3123 tests

Probability of success	0.001	0.01	0.11
Mean number of tests to accept		5889 tests	3753 tests
Mean number of tests to reject		5335 tests	2550 tests
Probability of success	0.002	0.00	0.00
Mean number of tests to accept		*	*
Mean number of tests to reject		3022 tests	1527 tests

- Since the probability of success is zero, the mean number of tests to accept is meaningless.

5. MODEL ELEMENTS: THE TEST-ANALYSE-AND-FIX APPROACH

Any reliability demonstration test procedure, whether for hardware or software systems, can either succeed or fail. If it fails, there is no use in testing another item, because it will probably suffer from the same design fault(s).

However, there is a significant difference between hardware reliability demonstration and software reliability demonstration. For hardware systems, the time required for analysing a failure and fixing the relevant fault(s) is relatively long. Consequently, failing a hardware reliability demonstration usually means, that the product will not be supplied on time, or that it will not meet its required reliability figure. On the other hand, for software systems the required time for analysing a failure and fixing the relevant fault(s) is relatively short. Therefore, while hardware reliability demonstration should be considered as a Go/No-Go or "One-Shot" procedure, software reliability demonstration should be considered as a multi-stage process: Test, Analyse, and Fix.

The Test-Analyse-And-Fix (TAAF) approach was adopted by the US Air Force as part of its R&M 2000 (Reliability and Maintainability) policy [9]. TAAF is a disciplined process for systematically detecting and eliminating design weaknesses while simulating the operational environment of hardware systems, in an iterative closed-loop manner. The TAAF process is a development strategy, and not a reliability demonstration testing, according to the specific contract.

Combining the TAAF approach with the SRST method for software reliability demonstration overcomes the single disadvantage of SRST. Whenever too many failures are found at the beginning of the test period, the SRST testing will be stopped, and will be resumed with a new software version. The SRST of the new version will not take into account the results of the previous tests, because it is a different product. The only problem now is, how to determine the best course of action, once a failure has been found: continue testing or fix the software.

Based on the SRST method and the TAAF approach, the optimal testing policy is a continuous decision rule. The testing policy uses (a) the number of missions testing with the current version of the software, (b) the number of failures found in this version, and (c) the time left till the deadline, to select one of the following options:

- Continue testing.
- Stop testing- the required reliability has been demonstrated.
- Stop testing- the reliability demonstration has failed.
- Stop testing, fix the software and start testing all over again later on, with a corrected version of the software. The testing of the new version starts from the beginning, and does not take into account the results of the previous versions.

In the typical testing policy of figure 2, the upper inclined line goes through all the break-even points (n, F) between the 'continue' option and the 'fix' option. In fact, it replaces the producer's risk line in the PRST method.

The only disadvantage of SRST in comparison with PRST is that for systems, which are not adequate, the number of tests could be larger than in PRST. This disadvantage is compensated by the above advantages, and may be eliminated altogether by incorporating the Test Analyse And Fix (TAAF) approach into SRST.

6. METHODS FOR IMPROVING THE CALCULATED RELIABILITY OR SHORTENING THE TESTING DURATION

6.1 Simultaneous testing of several systems

In cases when: (1) the mission duration is relatively long; (2) the reliability requirements, i.e. and/or c, are relatively high; and (3) the available time for reliability demonstration is relatively short, the required testing duration is simultaneous testing of N identical systems. Since all the systems have exactly the same software, n tests on one system are equivalent to n/N tests on each of N identical systems, carried at the same time. This means that the testing duration can be divided by N.

However, this method requires N systems, N oracles, N Mission-Profile Generators and N Monitors. In the aerospace domain the most problematic element is, of course, the avionics system itself, because of the following reason:

- Due to its severe environmental operating regime, an avionics systems is bound to cost much more than the other elements, which are not air-borne.
- The avionics system is, by definition, very special-purpose. On the other hand, the other elements. i.e. The Oracle, the Monitor and the Mission Profile Generator, are usually more general-purpose nature. This means that they may be used by other

projects, provided the appropriate software and hardware interfaces are replaced, and therefore their availability may be better.

Software reliability demonstration tests will typically take place before the beginning of mass production of the avionics system. Therefore the only available units at this stage are either prototypes or part of a very small pre-production batch. At this stage it makes no sense to produce more than a handful of units, for the following purposes:

1. Software reliability demonstration.
2. Hardware reliability demonstration, including Environmental stress screening (ESS) and Qualification tests (vibration, thermal. humidity, electrical, etc.).
3. Flight tests and narrow sample flights.

Hardware reliability demonstration tests examine the avionics system's endurance to its operational environmental regime throughout its lifetime. Some of the tests, such as EMP (Electro-Magnetic Pulse) and RFI (Radio Frequency Interference) endurance, may even be destructive. Therefore the units which have been subjected to these tests are not airworthy, and cannot be used for flight tests.

However, if the hardware qualification tests and ESS tests, which are usually much shorter than the software reliability tests (due to accelerated life-testing), are scheduled after the software reliability demonstration, then all the prototypes and pre-production units may be used for software reliability demonstration, which should be non-destructive. After this phase, some of the units would be assigned to ESS/Qualification tests, and some of them to flight tests/Narrow Sample flights.

The cost and the special-purpose nature of avionics systems will usually limit N to a very small number, which will very rarely be more than 10.

6.2 Accelerated testing

Another method of shortening the testing duration, which is much less straight-forward than simultaneous testing of N units, is accelerated software testing ([15], [16], [17]), which is based on the accelerated life testing of hardware systems.

In the software domain, the only possible analogue to “stress” is the input distribution to the software system. According to the software reliability demonstration model, each ball represents a whole mission cycle, comprising a trajectory of inputs to the system's software. Each trajectory comprises random data values, input sequence and time gaps between any two consequent inputs. The inputs to a system can be classified to the following categories:

1. Periodical inputs
   1.1. Discrete, digital or analogue inputs, which are periodically sampled by the software.
   1.2. Digital messages, which are periodically transmitted by external systems.
2. On-event inputs

2.1 Discrete, digital or analogue inputs, which are generated by external systems, including human operators, and interrupt the normal sequence of the software.

2.2 Digital messages, which are transmitted “on-event” by external systems, and interrupt the normal sequence of the programme.

The suggested procedure for accelerated software reliability testing requires:

- Increasing the periodical transmission frequency of digital messages by external systems.
- Increasing the frequency of on-event inputs from external systems.
- Increasing the frequency of on-event digital messages from internal systems.

All these changes may be implemented in the Mission Profile Generator.

Accelerated software reliability testing does not include increasing the frequency of periodical sampling by the system itself, because it would require changing the operational software itself.

When all these inputs are accelerated by the same factor, A, the accelerated mission profile comprises all the inputs which the system would encounter in a “regular” mission profile, in the correct sequence, but with smaller time gaps between the inputs.

Now instead of testing the software during n mission profiles, one may test it for only n/A mission profiles, and calculate the system's accelerated reliability according to the accelerated mission profile. In order to be able to adopt the accelerated reliability figure as a “normal” reliability figure, the following assumptions are required:

1. The real-time system would fail only to its inputs.
2. Subjecting a real-time system to more frequent inputs cannot improve its reliability.

The first assumption means, that without any external inputs, the software can work indefinitely, without any failure. In order to increase the confidence in this assumption, one could test one of the units without any external inputs, except for those which are absolutely needed for the system's operation. In order to demonstrate that the software is insensitive to the length of the mission, this unit may be tested by one very long continuous mission profile. It should never fail.

The second assumption means, that more frequent inputs tend to “stress” the software, and hence the probability of failure can either remain constant or increase, but it can never decrease. It also means, that once the software has been subjected to a trajectory of inputs, which cause it to fail, the prompt arrival of new inputs cannot prevent its failure. In order to increase the confidence in this assumption, one could test one of the units with non-accelerated inputs, and compare the results of both kinds of tests. The results of the non-accelerated test should be at least as good as the results of the accelerated tests, at a confidence level corresponding to c (the required confidence level in 1-θ). Therefore, the result of accelerated software reliability tests is a conservative estimate of the real software reliability.

6.3 Practical limitations in the aerospace domain

1. Any avionics system's hardware cannot deal with periodical inputs whose frequency is above a certain threshold. This constraint is more typical of digital input messages.
2. The Mission Profile Generator, being based on a more user-friendly hardware and software than the avionics system, may have even more severe limitations on the maximum frequency of simulated inputs to the avionics system.

Due to the characteristic short time factors associated with airborne platforms, it would be very rare to achieve an accelerating factor in excess of 10 in any real-time avionics system. However, using accelerated testing as well as simultaneous testing of several units, one may shorten the testing duration by a factor of up to 100, which is quite significant.

6.4 Parallel versions

The third method of shortening the duration of software reliability demonstration has more to do with increasing the probability of success than with actually shortening the testing duration.

Whenever the penalty for failing to demonstrate the required reliability is large enough, and the number of allowable failures is at least one, it may be worthwhile to assign two or more teams of software professionals to the project once a failure is found. While the first team continues to test the first version of the software, trying to demonstrate its reliability with the current number of failures, the second team produces a new version, and starts testing it from the beginning. Now there are two parallel versions, which are simultaneously tested. When a failure is found in either version, a third version may be created and tested in parallel, etc. This method requires at least two units of the tested systems, and at least two complete testing set-ups. The labour cost of this option is, of course, much higher than in the case of a single version. Moreover, the development of several parallel software versions in this manner represents a challenge to the software configuration control programme. Therefore it may be justified only in certain cases, depending on the penalty cost.

6.5 Using additional information

In certain circumstances, the Error Seeding Model may increase the confidence in the results of the reliability demonstration testing, both quantitatively and qualitatively. Let us assume that the reliability demonstration testing is carried out independently on two different versions: the original version (installed in one of N units), and a “seeded” version, installed in

only one unit. The seeded version contains s known errors, each of them leading to failures, given the right trajectories.

In order to use the Error Seeding Model, the seeded errors have to represent the unknown errors, such that the probabilities of finding both kinds of errors would be similar. The best way of doing it is not to "invent" the seeded errors, but to use actual errors, which have been discovered and corrected during the software development and testing process.

Let us assume, that at the end of the reliability demonstration testing, the required reliability of the original version, θ_1, has been demonstrated with the required confidence level, c, without any failure. At the same time, the testing of the "seeded version" has revealed $j \leq s$ of the seeded faults.

According to the TRW formula, the probability that the reliability is smaller than θ_1 is at least c. On the other hand, according to the error seeding model, the probability that the seeded version does not contain any indigenous faults is at least $j/(s+1)$. But if the seeded version has no indigenous faults then neither does the original version, which means that its reliability is 0. Hence, whenever $j/(s+1)>c$, one can conclude that the confidence level in θ_1 is not c, but $j/(s+1)$, which is greater. Obviously, s must be a few dozens and j must be very close to s in order for this inequality to apply. For example, for $\theta_1=0.001$, n=3,000 and s=29, the calculated c is 0.95. If at the same time, all the 29 seeded errors have been found (j=29), then c can be increased to 29/30=0.966, which is a little better.

When j=s, the confidence in the results of the reliability demonstration testing is increased even further, though only qualitatively, because it indicates that both the testing process (MPG and Oracle) and the number of tests are very effective.

The Error Seeding method has another useful benefit: it may have a positive psychological effect on the testing effort, especially prior to the reliability demonstration phase [18].

6.6 Necessary conditions for using the model

6.6.1 Reliability requirements and mission duration

6.6.1.1 Single unit, non-accelerated testing

The total time required for reliability demonstration with no failures is given by:

$T_0=n_0T_{ave}=[\ln(1-c)]/[\ln(1-\theta_1)]T_{ave}$.

Therefore T_{req} increases for larger values of c, smaller values of θ_1 and larger values of T_{ave}. In practice, the total time assigned to software reliability demonstration of a safety-critical avionics system may be several months. It would be longer if the tests continue throughout the flight tests period (from a few weeks to a few months), and longer still if they continue throughout the Narrow Sample period (a few months). The total time available may be calculated using the following assumptions:

1. The maximum calendar time available is 6 months until the flight tests begin and 12 months until the system is operationally deployed in the aircraft.
2. The tests are fully automated and are carried out 24 hours a day, 7 days a week.

Hence the total time available for testing is between 4,380-8,760 hours.

Since the typical mission duration of a modern fighter aircraft, without refuelling, is one hour, then without using several units and accelerated testing, the maximum number of tests is 4,380 or 8,760. Using the TRW formula, the smallest unreliability values which can be demonstrated for common values of c, with no failures, are as follows:

C	0.90	0.95	0.99	0.995	0.999
θ_1	0.00026	0.00034	0.00052	0.00060	0.00079

For the typical value of θ_1=0.001, the maximum number of failures can be:

C	0.90	0.95	0.99	0.995	0.999
F_{max}	4	3	2	1	0

For T_{ave}=2 hours, for instance, the required testing duration is doubled, and the minimum values of θ_1 are also doubled.

6.6.1.2 Several units, accelerated testing

Testing N units and accelerating the tests by a factor of A, the total time required is given by:

$T_{req}=[\ln(1-c)]/[\ln(1-\theta_1)]T_{ave}/AN$.

Clearly, T_{req} decreases for larger values of A and N.

Using only 5 units and accelerating the tests by a factor of 2, one can either demonstrate the same reliability with a 10 times shorter test duration, or demonstrate 10 times lower reliability requirements with the same test duration. In these circumstances, it would be practical to require every 10^{-5} as the target unreliability.

6.6.2 Correction time, reliability requirements and mission duration

The model can only be used when T_{corr}, the time needed for producing a new software version, is smaller than the difference T_{max}-T_0 (a function of θ_1, c and T_{ave}). Otherwise there is no point in fixing the software, for there would not be enough time left for testing the new version.

7. WHEN IS THE BEST TIME FOR A RELIABILITY/DEPENDABILITY DEMONSTRATION?

7.1 The various options

Software Dependability Demonstration (SDD) is usually going to take a few months. (For example, a failure-free demonstration of 0.999 dependability with 95% confidence would require 3,000 mission cycles). Since it is a prerequisite for the full-scale deployment of a safety-critical software system, the scheduling of software dependability demonstration testing within the software development process is very important. Software dependability demonstration must never begin before the successful completion of all the deterministic software testing, usually by the software development team, and of all the hardware testing. From this stage the software system must successfully complete three different activities:

- Software dependability demonstration.
- Flight tests.
- Narrow sample flights: (After flight testing the new operational software system and before installing it in all the relevant aircraft, the software is installed in a small number of operational aircraft for a predetermined period of time (several months). During this time this software is experienced by many operational air-crews, sometimes with a few operational limitations. The purpose of the narrow sample flights is to locate any remaining faults, and to approve or disapprove of installing the new software in all the aircraft).

The scheduling of these activities must obey the following constraints:

- Narrow sample flights cannot begin before the successful completion of flight tests, and it is always the last activity prior to full-scale deployment. The main reason for scheduling flight tests before the narrow sample flights is that the software may contain embarrassing or even dangerous faults, and therefore should be tested first by experienced test pilots under controlled conditions. However, flight tests cannot replace narrow sample flights, because they are very expensive, while narrow sample flights are virtually free (they are part of the routine flights of operational squadrons).
- Software dependability demonstration may take place before flight tests, after flight tests or in parallel with flight tests; and before or in parallel with narrow flights.

In view of the above constraints there are 5 possible ways of scheduling these 3 activities:

1. Software Dependability Demonstration, Flight Tests, Narrow Sample.
2. Flight Tests, Software Dependability Demonstration, Narrow Sample.
3. Flight Tests, Software Dependability Demonstration in parallel with Narrow Sample.
4. Flight Tests in parallel with Software Dependability Demonstration, Narrow Sample.
5. Software Dependability Demonstration in parallel with Flight Tests and later with Narrow Sample.

7.2 Guidelines to the optimal option according to various criteria

The scheduling of SDD, flight tests and narrow sample flights may be subject to various constraints, such as time to operational deployment, total budget and the number of available units. It may also be optimised according to various criteria, part of which are listed below.

- Minimum total time till operational deployment

According to this "success-oriented" approach, the best options are option3 and option 5, because both of them perform the lengthy SDD in parallel with the other long phase, i.e. narrow sample flights. However, both methods need a larger number of units, and their probability of success at the first attempt is relatively low. In order to choose between these two options one has to compare the benefit of the lower probability of failing the SDD in option 3 with the benefit of the lower probability of failing the narrow sample flights in option 5.

- Minimum number of units

When this is the paramount consideration, one has to choose among options 1 and 2, since option is clearly inferior, both in terms of its probability of success and its minimum total time. Usually option 1 should be preferred to option 2, since the higher probability of success of its flight tests phase means lower expected total cost than in option 2.

- Minimum expected cost

Since the most expensive phase by far is the flight tests phase, the best option according to this criterion is option 1. In all the other options, the flight tests phase is the first phase in the schedule.

- Maximum probability of success at first trial

The best options according to this criterion are option 1 and 2. In order to select between these two options one should estimate the probabilities of success of each of the three phases in both options.

- Minimum expected time

Since the SDD and the narrow sample phases are much longer than the flight tests phase, the best option according to this criterion would be option 2, in which the probabilities of success of these phases is higher.

9. A GENERAL TESTING STRATEGY

An optimal testing policy is a testing policy, which yields the minimum testing time or cost till success, or the maximum probability of success.

A safety-critical system is planned to undergo statistical testing in order to demonstrate its reliability in terms of (θ_1, c), i.e. that the probability that its unreliability is larger than θ_1, is small than 1-c. The time allocated to the reliability is T_{max} hours. In some cases, $T_{max}=\infty$ and the maximum allowable number of failures is zero. This situation can arise, for example,

when a safety-critical system is scheduled to go on its first operational mission immediately after its reliability has been demonstrated. T_{max} hours correspond to n_{max} mission cycles, of T_{ave} duration time: $n_{max}=T_{max}/T_{ave}$.

According to the SRST reliability demonstration method, this requirement can be met by a infinite number of pairs (n,F), with n mission cycles and F failures found, according to the following formula:

$$1-\sum_{j=0}^{F} nCj.\ (1-\theta_1)^{n-j}\ .\theta_1^{\ j} \geq c \qquad F=0,1,2....F,\ n\leq n_{max}$$

The required number of tests for any number of failures, F, is the largest integer, n_F, satisfying this inequality.

The maximum allowable number of failures during the demonstration, F_{max}, is the smallest F such that: $T_{F+1}=n_{F+1}T_{ave}\geq T_{max}$

Sometimes there may be a specific requirement to demonstrate the reliability with zero failures, that is $F_{max}=0$. In this case the required number of tests is given by:

$n_0=\ln(1-c)/\ln(1-\theta_1)$

This requirement may or may not be combined with unlimited testing time.

Clearly, n_{max} must be greater than n_0 and T_{max} must be greater than T_0, otherwise the required reliability cannot be demonstrated.

Combining the TAAF approach to the SRST method, the reliability demonstration testing is composed of one or more phases, where in each phase a different software version is tested. n_F is the required number of tests with no more than F failures in any phase of the reliability demonstration testing. The software reliability demonstration testing can therefore be regarded as a process:

- Test the current version to demonstrate its reliability. If no failures are found during T_0 hours, stop testing.
- When the Fth failure is found (F=1,2,…,F_{max}), choose between the two following options:

1. Correct the current version to produce a new version and start testing all over again. This option requires that for the new version $T_{max}\geq T_0+T_{corr}$
2. Continue testing, aiming at T_F hours with only F failures, F=1,2…F_{max}.

Because of the time constraint (T_{max}), this process can either succeed or fail.

The maximum number of new versions when any failure is found t hours before the deadline, can be calculated as follows:

Assuming that there is enough time for at least one more version, $t-T_{corr}$ is larger than T_0, and the testing of the first new version will start T_{corr} hours after finding the failure. If the first failure in the new version is found immediately, then the testing of the second new version will start at least $2T_{corr}$ hours after finding the original failure. Similarly, the testing of the Vth new version will start at least VT_{corr} hours after finding the original failure. The only

constraint is that the time left, t-VT_{corr}, is still larger than T_0. Therefore, the maximum number of new versions is the integer solution of:

$t-V_{max}T_{corr} \geq T_0$

$V_{max}=int[(t-T_0)/T_{corr}]$

Sometimes a penalty cost, P_c, may apply if the reliability demonstration fails. The penalty cost can be, for example, the cost of a lost contract or tender, or a specific penalty payment on failing to supply the software with the required reliability on time.

8.3.5 Optimal testing policies

The model enables one to calculate the optimal testing policies in various cases, such as:

1. $F_{max}=0$, $T_{max}=\infty$: the system is required for an operational mission, whose dates hasn't been determined yet. The intuitive policy is to stop testing after the first failure, whenever it happens, and fix the fault.
2. $F_{max}=0$, $T_{max}\neq\infty$: the software is required for an operational mission, whose date has been determined. In this case there is no intuitive policy,
3. $F_{max}\geq 0$, $T_{max}\neq\infty$ and the software is required for operational use, but not for a specific operational mission. This is the most general case of the model. Again, there is no intuitive policy.
4. $F_{max}\geq 0$, $T_{max}\neq\infty$ and there is a financial penalty, P_c, on failing to demonstrate the required reliability. In this case, the optimal policy may be parallel development of two or more software versions.

Formulation and solution of these optimisation problems using the SRST and TAAF approach is based on the notion of break-even points. For every failure found, the optimal policy is to continue testing if the time left is smaller than the break-even point value, and stop testing if the time left is larger than this value. The values of the various break-even points can be found by stochastic dynamic programming and are discussed in [19].

REFERENCES

[1] US Department of Defense. *MIL-HDBK-781A: Handbook for Reliability Test Methods, Plans and Environments for Engineering, Development Qualification, and Production.* 1996.

[2] I.D. Hill. *An Introduction to Sampling Inspection.* Monograph 62/1. The Institution of Engineering Inspection. 1960.

[3] US Government Printing Office. *MIL-STD-105D. Sampling Procedures and Tables for Inspection by Attributes.* 1958.

[4] British Standards Institute. *BS6001: Parts 0-3 (ISO 2859) Sampling Procedures for Inspection by Attributes.* 1996.

[5] O. Tal, A. Bendell and C. McCollin. Short Communication A Comparison of Methods for calculating the duration of software reliability demonstration testing, particularly for safety-critical systems. *Quality and Reliability Engineering International.* Issue 16 2000. pp 59-62.

[6] O. Tal. Software Dependability Demonstration for Safety-critical Military Avionics Systems by Statistical Testing. Ph.D. Thesis. The Nottingham Trent University. 1999.

[7] M. A. Friedman, J. M. Voas. *Software Assessment: Reliability, Safety, Testability.* Wiley. 1995.

[8] T.A. Thayer, M. Lipow, E.C. Nelson. *Software Reliability*. North Holland, 1978.

[9] USAF. Reliability and Maintainability 2000. 1987.

[10] K.S. Trivedi, Probability and Statistics. 1982, Prentice-Hall Inc.

[11] N.R. Mann, R.E. Schafer, N.D. Singpurwalla. *Methods for Statistical Analysis of Reliability and Life Data*. Wiley, 1974, pp297-303.

[12] B. Epstein, M. Sobel. Sequential Life Tests in the Exponential Case. *Annals of Mathematical Statistics*, vol. 26, March 1953, pp 82-95.

[13] A. Wald. *Sequential Analysis.* Wiley. 1947.

[14] B.K. Ghosh. *Sequential Tests of Statistical Hypotheses*. Addison-Wesley 1970.

[15] P. Thevenod-Fosse, H. Waeselynck. On Software Dependability Evaluation from a Statistical Testing Approach. *PDCS Technical Report Series Series no. 28*, July 1990.

[16] H. Zhu. Software Testing via Environment Simulation. *CONTESSE Task 4, Version 2*, July 1994.

[17] M. S. Alam, W. H. Chen, W.K. Ehrlich, M.E. Engel, D.A. Kropfl, P. Vema. Assessing Software Reliability Performance under Highly Critical but Infrequent Event Occurrences. *Proceedings IEEE International Symposium on Software Reliability Engineering - Case Studies.* Albuquerque, 1997, pp294-307.

[18] G.J. Myers. *The Art of Software Testing.* Wiley. 1976.

[19] O. Tal, C. McCollin and A. Bendell. An optimal statistical testing policy for software reliability demonstration of safety-critical systems. *European Journal of Operational Research 2/6/99.* Awaiting referee's comments.

AUTHORS

Dr. Oded Tal; RAFAEL P.O. Box 2082, Haifa 31021 Israel.

Oded Tal was born in 1957. He received the BSc (1979) from The Technion and an MSc (1990) from Tel-Aviv University. He worked in Rafael from 1984-1996. He received his PhD from The Nottingham Trent University in 1999 and has now returned to work for RAFAEL.

Dr. Chris McCollin; Department of Mathematics, Statistics and Operational Research and The Quality Unit, Nottingham Trent University, Burton Street; Nottingham NG1 4BU UK.

Chris McCollin was born in 1955. He received the BSc (1976) from the University of Newcastle upon Tyne, an MSc (1980) from the University of Birmingham and PhD (1993)

from Nottingham Trent University. He has worked as a reliability engineer for various large aerospace companies. Currently he is a senior lecturer at the Quality Unit, Nottingham Trent University.

Professor Tony Bendell; The Management Centre and Department of Engineering, University of Leicester, University Road, Leicester LE1 7RH.

Tony Bendell was born in 1949. He received the BSc (Econ) (1971) and the MSc (1972) from London University and PhD (1982) from CNAA. From 1984 to 1999, he was the Head of the Department of Mathematics, Statistics and Operational Research and the Director of the Quality Unit at Nottingham Trent University. He is now Professor of Quality and Reliability Management at the University of Leicester.

Figure 1: A Simplified Oracle

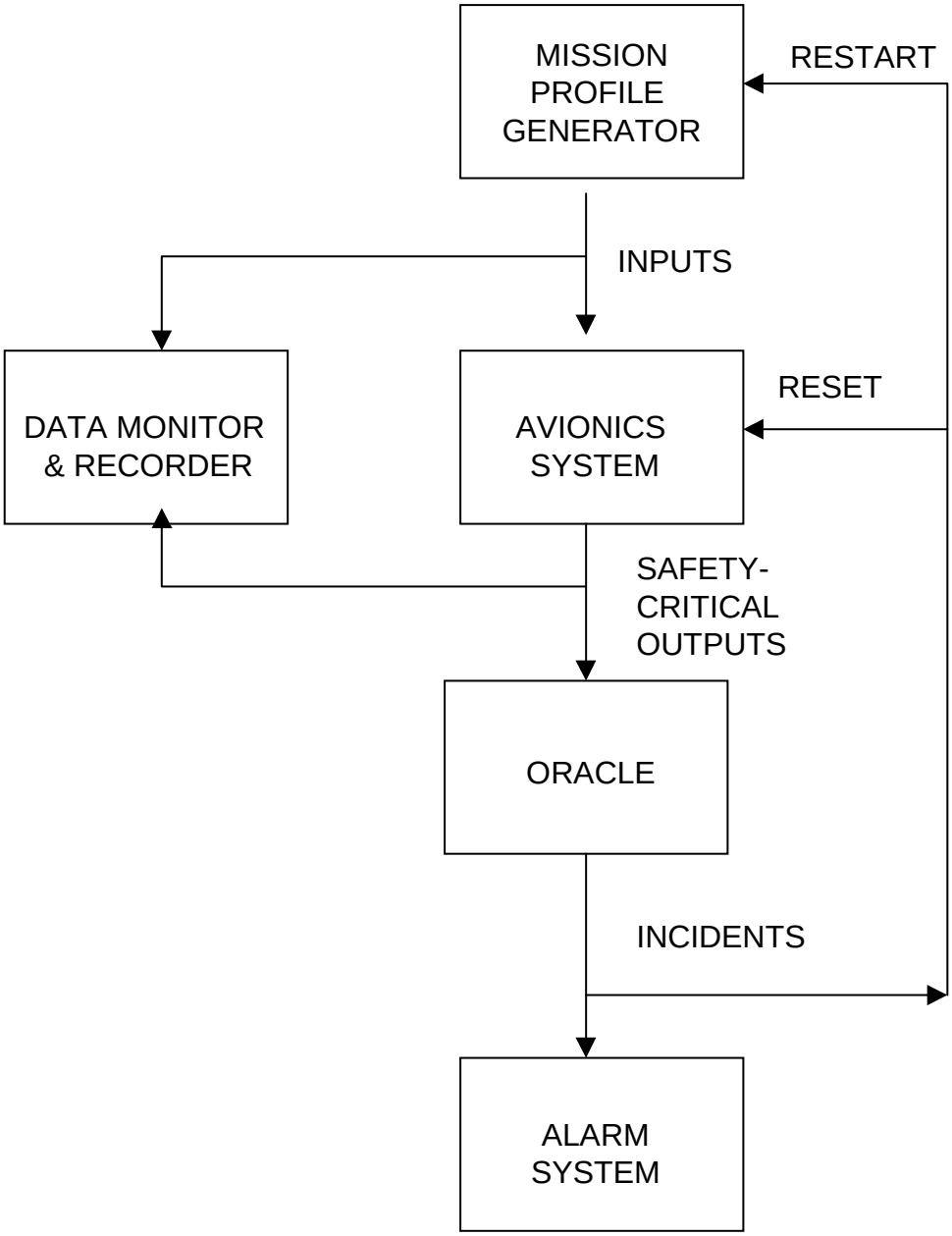

Figure 2. A graphical decision rule for SRST

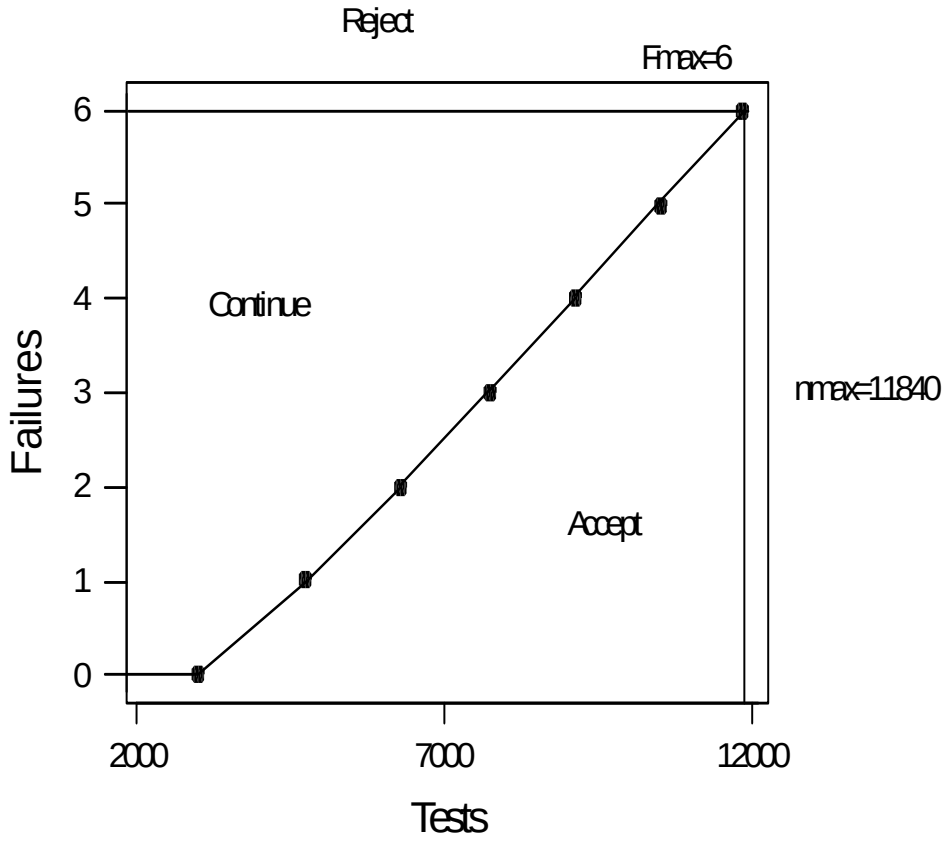

Figure 3. A graphical decision rule for PRST