Another instance in which a reliability projection model would be useful is when the current test phase contains a number of design configurations of the units under test due to incorporation of reliability fixes during the test phase. If there is a lack of fit of the reliability growth tracking model as a result of these differing configurations, then a tracking model should not be used to assess the reliability of the latest configuration, or for extrapolation to a future milestone. Such a lack of fit may be due to the timing of the corrective action process (i.e., when the fixes are implemented) and their associated effectiveness (as defined by the FEF). As pointed out earlier, the AMPM, unlike a tracking model, is insensitive to any "non-smoothness" in the expected number of failures versus test time that results from the timing or fix effectiveness of corrective actions. In such a situation, program management may wish to use a projection method such as the AMPM to assess the reliability of the current configuration, or to project the expected reliability at a future milestone.

The AMPM can also be used to construct a useful reliability maturity metric. This metric is the fraction of the expected initial system B-mode failure intensity surfaced by test duration, t.

Table 3.6.2.2-1 summarizes the options, required inputs and calculated outputs associated with AMPM.

Table 3.6.2.2-1: AMPM Reliability Growth Projection Model Options, Required Inputs and Calculated Outputs

Options	**Required Inputs**	**Calculated Outputs**

• Compute estimates for (1) B-Mode initial failure intensity, (2) expected number of B-Modes surfaced, (3) percent surfaced of the B-mode initial failure intensity, (4) projected failure intensity, and (5) projected MTBF
• <u>**Option 1**</u>: Individual B-Mode First Occurrence Time Data
  ○ **Sub-option 1A:** Single FEF Method
    ▪ **Sub-option 1A1:** All fixes delayed
      • <u>Case A:</u> Repeating B-Modes
      • <u>Case B:</u> No B-Mode Repeats
    ▪ **Sub-option 1A2:** Not all fixes delayed
  ○ **Sub-option 1B:** Gap Method
  ○ **Sub-option 1C:** Segmented FEF Method
• <u>**Option 2**</u>: Grouped Data approach
	<u>Option 1, 1A, 1A1, Case A:</u>
	• Total Test Time
	• Number of A-Mode Failures
	• Number of Observed B-Modes
	• Number of Projections to Make
	• Initial (Assumed) Number of B-Modes
	• Total Number of B-Mode Failures (First occurrences and repeats)
	• Average B-Mode FEF (if not entered separately for each B-Mode, below)
	• **For <u>each</u> B-Mode:**
	  ○ First occurrence time
	  ○ Individual FEF (Optional)
	• **For <u>each</u> Projection:**
	  ○ Time at which Projection is Made
	• **Depending on Plot to be Generated:**
	  ○ Total Test Time
	  ○ Start/Stop Test Times
	  ○ Number of Groups
	<u>Option 1, 1A, 1A1, Case B:</u>
	• Same as Option 1, 1A, 1A1, Case A
	<u>Option 1, 1A, 1A2:</u>
	• Same as Option 1, 1A, 1A1, Case A, <u>except</u>
	  ○ No Case A or Case B Option
	<u>Option 1, 1B:</u>
	• Same as Option 1, 1A, 1A1, Case A, <u>except:</u>
	  ○ Replace "Initial (Assumed)
		<u>Option 1, 1A, 1A1, Case A:</u>
		• Average FEF for the B-Modes
		• Estimate of A-Mode Failure Rate
		• Estimate of MTBF Growth Potential (based on Finite Number of Initial (Assumed) B-Modes)
		• Estimate of MTBF Growth Potential (based on Infinite Number of Initial (Assumed) B-Modes)
		• Estimate of Initial B-Mode Failure Intensity (based on Finite Number of Initial (Assumed) B-Modes)
		• Estimate of Initial B-Mode Failure Intensity (based on Infinite Number of Initial (Assumed) B-Modes)
		• Estimate of Reliability Growth Parameter (based on Finite Number of Initial (Assumed) B-Modes)
		• Estimate of Reliability Growth Parameter (based on Infinite Number of Initial (Assumed) B-Modes)
		• Estimate of Model Scale Parameter
		• Smallest Integer for the Initial (Assumed) Number of B-Modes for which the Model Exists
		<u>Option 1, 1A, 1A1, Case B:</u>
		• Same as Option 1, 1A, 1A1, Case A
		<u>Option 1, 1A, 1A2:</u>
		• Same as Option 1, 1A, 1A1, Case A
		<u>Option 1, 1B:</u>
		• Estimate of A-Mode Failure Rate
		• Gap Size
		• Estimate of Reliability Growth Parameter
		• Estimate of Initial B-Mode failure Rate
		• Estimate of Failure Rate Growth Potential
		• Estimate of MTBF Growth Potential

Options	Required Inputs	Calculated Outputs
	Number of B-Modes" with "Endpoint for the Gap" Option 1, 1C: • Same as Option 1, 1A, 1A1, Case A, except: ○ Replace "Initial (Assumed) Number of B-Modes" with "A Partition Point Less Than the Total Test Time" ○ Average B-Mode FEF (if not entered separately for each B-Mode) Before the Partition Point ○ Average B-Mode FEF (if not entered separately for each B-Mode) After the Partition Point Option 2: ○ Total Test Time ○ Number of Observed B-Modes ○ Number of A-Mode Failures ○ Number of Projections to Make ○ Number of Groups ○ Enter Time Value where Projected MTBF is to be Computed • **For each Group:** ○ Test Time ○ Number of New B-Modes • **For each B-Mode:** ○ First Occurrence Time ○ Individual FEF (Optional) • **For each Projection:** ○ Time at Which Projection is Made • **Depending on Plot to be Generated:** ○ Total Test Time ○ Start/Stop Test Times	• Number of B-Modes Excluded by Jumping the Gap Option 1, 1C: • Estimate of A-Mode Failure Rate • Average FEF Before the Partition Point • Average FEF After the Partition Point • Estimate of Initial B-Mode Failure Rate • Estimate of Reliability Growth Parameter • Estimate of Failure Intensity at the Partition Point • Estimate of Failure Intensity Growth Potential • Estimate of MTBF Growth Potential Option 2: • Average FEF for the B-Modes • Estimate of A-Mode Failure Rate • Estimate of Reliability Growth Parameter • Estimate of Initial B-Mode Failure Rate • Estimate of Rate of Occurrence of New B-Modes at Total Test Time • Projected Failure Intensity at Total Test Time • Failure Intensity Growth Potential • Projected MTBF at Total Test Time • MTBF Growth Potential • Projected MTBF at User-Input Time Value

The relevant equation for the AMPM system failure intensity (after fixes to all B-modes surfaced by test time, t, have been implemented) is:

$$r(t;\underline{\lambda}) = \lambda_A + \sum_{i=1}^{K} \lambda_i - \sum_{i=1}^{K} d_i \lambda_i I_i(t)$$

The key AMPM reliability projection parameters in terms of K, and the gamma distribution parameters of α and β are:

- The expected/estimated value of the sum of the B-mode random sample size gamma variables for both finite and limitless conditions:

$$\lambda_{B,K} = K\beta(\alpha + 1)$$

$$\hat{\lambda}_K = K\hat{\beta}_K(\hat{\alpha}_K + 1)$$

$$\hat{\lambda}_\infty = \frac{m\hat{\beta}_\infty}{\ln\left(1 + \hat{\beta}_\infty T\right)}$$

- The expected/estimated number of distinct B-modes at time, t, for both finite and limitless conditions:

$$\mu(t) = K\left[1 - (1 + \beta t)^{-(\alpha+1)}\right]$$

$$\hat{\mu}_K(t) = K\left[1 - \left(1 + \hat{\beta}_K t\right)^{-(\hat{\alpha}_K+1)}\right]$$

$$\hat{\mu}_\infty(t) = \left(\frac{\hat{\lambda}_{B,\infty}}{\hat{\beta}_\infty}\right)\ln\left(\left(1 + \hat{\beta}_\infty t\right)\right)$$

- The unconditional expected/estimated B-mode rate of occurrence at time, t, for both finite and limitless conditions:

$$h(t) = \frac{\lambda_{B,K}}{(1 + \beta t)^{\alpha+2}}$$

$$\hat{h}_K(t) = \frac{\hat{\lambda}_K}{\left(1 + \hat{\beta}_K t\right)^{\hat{\alpha}_K+2}}$$

$$\hat{h}_\infty(t) = \frac{\hat{\lambda}_{B,\infty}}{1 + \hat{\beta}_\infty t}$$

- The expected/estimated value of the system failure intensity and growth potential with respect to first occurrence time of the B-modes for both finite and limitless conditions:

$$\rho(t) = \lambda_A + (1 - \mu_d)\lambda_{B,K} + \frac{\mu_d\lambda_{B,K}}{(1 + \beta t)^{\alpha+2}}$$

$$\hat{\rho}_{GP,K} = \hat{\lambda}_A + \left(1 - \mu_d^*\right)\hat{\lambda}_{B,K}$$

$$\hat{\rho}_{GP,\infty} = \hat{\rho}_{GP,\infty} + \left(1 - \mu_d^*\right)\hat{\lambda}_{B,\infty}$$

$$\hat{\rho}_K(t) = \hat{\rho}_{GP,K} + \mu_d^*\hat{h}_K(t)$$

$$\hat{\rho}_\infty(t) = \hat{\rho}_{GP,\infty} + \mu_d^*\hat{h}_\infty(t)$$

- Expected/estimated fraction of $\lambda_{B,K}$ surfaced as a function of time, t, for both finite and limitless conditions:

$$\theta(t) = 1 - (1 + \beta t)^{-(\alpha+2)}$$

$$\hat{\theta}_K(t) = 1 - \left(1 + \hat{\beta}_K t\right)^{-(\hat{\alpha}_K + 2)}$$

$$\hat{\theta}_\infty(t) = \frac{\hat{\beta}_\infty t}{1 + \hat{\beta}_\infty t}$$

<u>Example</u>

The discussion in MIL-HDBK-189A, Section 7.6.7, illustrates several key features of the AMPM and associated estimators by applying the model to a data set generated during an Army system development program. For this specific example, only the B-modes were considered and the failure intensity of the A-modes of the dataset is set to zero. The test data consists of 163 B-mode first occurrence times (i.e., there is a total of 163 unique B-modes) generated over 8000 "equivalent" mission hours.

Figure 3.6.2.2-2, displays the cumulative number of distinct B-modes versus cumulative mission hours. The graph also illustrates the estimate of the expected number of B-modes, $\hat{\mu}_K(t)$, for several values of the potential number of unique B-mode occurrences (K), generated over time, t, both seen and unseen.

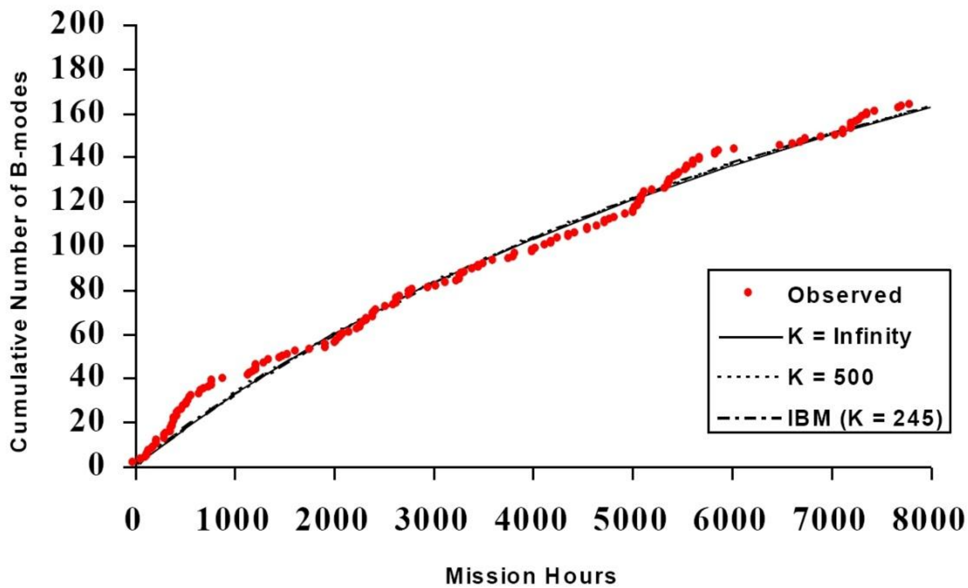


Figure 3.6.2.2-2: Observed versus Estimate of Expected Number of B-Modes as a Function of K

Figure 3.6.2.2-3 illustrates the extrapolation of the expected number of B-modes as a function of K. Note that the actual data ends at 8000 hours. The extrapolations cover from 8000 hours to 30,000 hours.

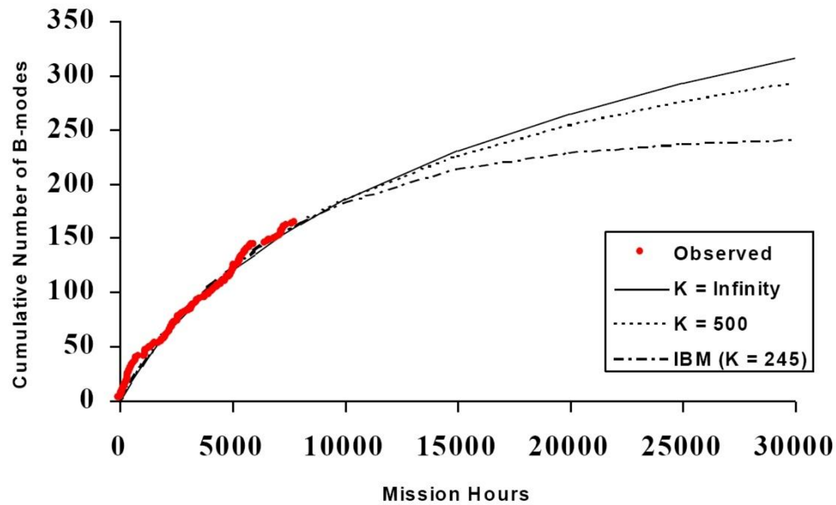


Figure 3.6.2.2-3: Extrapolation of Estimated Expected Number of B-Modes as a Function of K

Figures 3.6.2.2-4 and 3.6.2.2-5 present extrapolations for the projected MTBF and estimated fraction of expected initial B-mode failure intensity, respectively. The graph of projected MTBF is based on an average FEF of 0.70 and an assumed failure rate of zero for the A-modes.

In the interpretation of Figure 3.6.2.2-4, the model based on "K = Infinity" appears to provide a more conservative estimate of the projected MTBF than any of the other K estimators, as one might expect. MIL-HDBK-189A makes the point, however, that for values of "t" greater than the actual 8000 mission hours, the values for the expected number of B-modes (Figure 3.6.2.2-3), the projected MTBF (Figure 3.6.2.2-4) and the estimated fraction of expected initial B-mode failure intensity (Figure 8.5-4) quickly become much closer to the "K = Infinity" graph than to the "K = K_{IBM}" graph as K increases above the K_{IBM} value.

It can also be observed from Figure 3.6.2.2-5, that the estimated fraction of expected initial B-mode failure intensity approximately equals 0.67 over the range K_{IBM} to "K = Infinity". Therefore, regardless of the "true" value for K, it is estimated that the remaining B-modes contribute about $(0.33)(\lambda_B)$ to the overall system failure intensity.

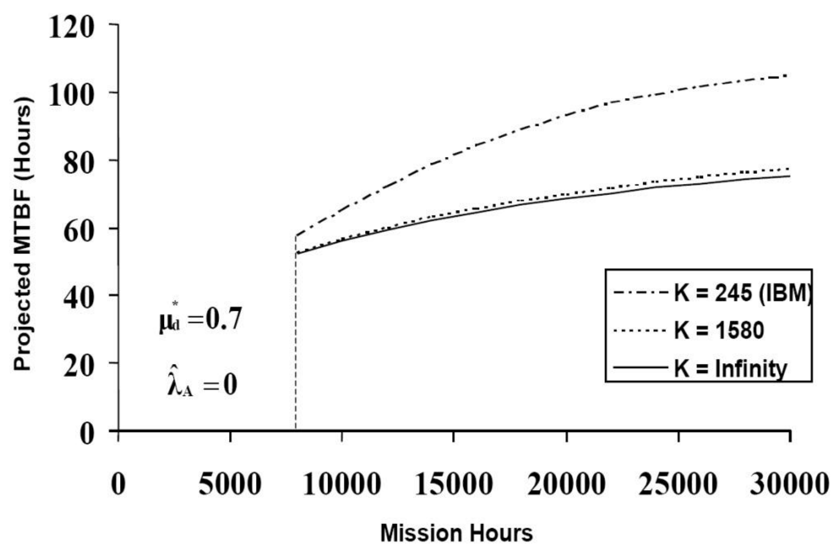


Figure 3.6.2.2-4: Projected MTBF for Different K Values (Based on Initial 8000 Hours of Test Data)

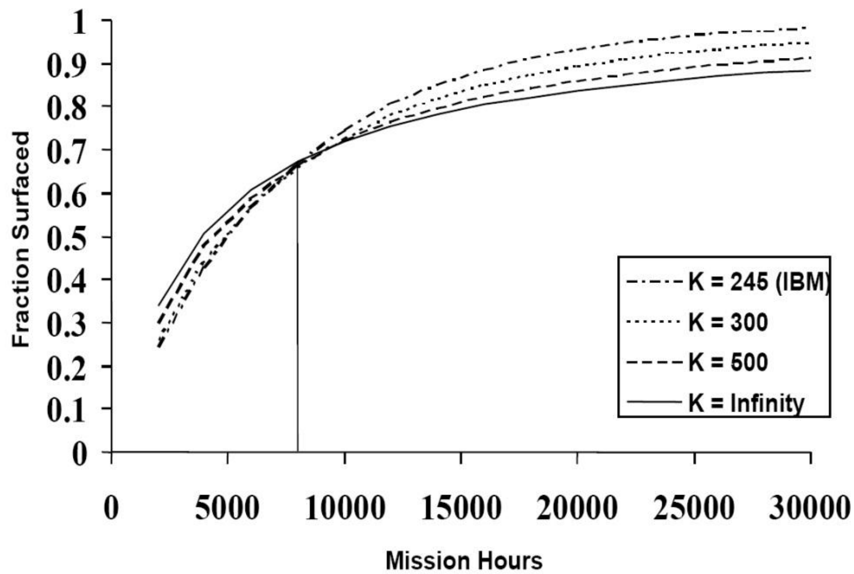


Figure 3.6.2.2-5: Estimated Fraction of Expected Initial B-Mode Failure Intensity Surfaced for Different K Values (Based on Initial 8000 Hours of Test Data)

For More Information:

1. Nicholls, D., P. Lein, T. McGibbon, “Achieving System Reliability Growth Through Robust Design and Test”, Reliability Information Analysis Center, 2011.
2. MIL-HDBK-189, “Reliability Growth Management”, 13 February 1981
3. MIL-HDBK-189C, “Reliability Growth Management”, 14 June 2011

Topic 3.6.2.3: Software Reliability Growth Models

Formal reliability growth testing for software, similar to that for hardware, is performed to measure the current reliability, identify and eliminate the root cause of software faults and forecast future software reliability. Software reliability growth testing should always be performed under the same operational profiles as those expected in the field in order to be effective.

There are, literally, hundreds of software reliability growth, prediction and estimation models available. In order to accurately and effectively measure and project reliability growth requires the use of an appropriate mathematical model that describes the variation of software reliability behavior over time. Parameters for these growth models can be obtained either from Design for Reliability analyses and testing performed during the time period that precedes formal reliability growth testing, or from estimations performed during the test. Table 3.6.2.3-1 provides a summary of characteristics of some of the most common software reliability models (see Reference 1 for additional details).

Table 3.6.2.3-1: Summary of Software Reliability Models

Model Name	Hazard Function Formula	Required Data or Estimation	Limitations and Constraints
General Exponential (general form of the Shooman; Jelinski-Moranda; and Keene-Cole exponential models)	$\mathbf{z(x) = K[E_0 - E_c(x)]}$	• Number of corrected faults at some time, x (E_c) • Estimate of initial number of faults that will lead to failure (E_0) • Failures per time unit, per faults remaining (K)	• Software must be operational • Assumes no new faults are introduced during corrective action • Assumes linear reduction in number of residual faults over time
Musa Basic	$\lambda(\mu) = \lambda_0 \left[1 - \frac{\mu}{\upsilon_0}\right]$	• Number of detected faults at some time, μ • Estimate of initial number of faults that will lead to failure (λ_0) • Estimate of number of failures that would occur over infinite time (ψ_0)	• Software must be operational • Assumes no new faults are introduced during corrective action • Assumes linear reduction in number of residual faults over time
Musa Logarithmic	$\lambda(\mu) = \lambda_0 e^{(-\phi\mu)}$	• Number of detected faults at some time, μ • Estimate of initial number of faults that will lead to failure (λ_0) • Relative change of failure rate over time (ϕ)	• Software must be operational • Assumes no new faults are introduced during corrective action • Assumes exponential reduction in number of residual faults over time
Littlewood/ Verrall	$\lambda(t) = \frac{\alpha}{[t + \Psi(i)]}$	• Estimate of number of failures, α • Estimate of reliability growth, X(i) • Time between failures detected or the time of failure occurrence, t	• Software must be operational • Assumes uncertainty in the corrective action process (fixes may introduce defects, improvements are of uncertain magnitude)
Schneidewind	$\mathbf{d_i = \alpha e^{(-\beta i)}}$	• Faults detected in equal time interval, i • Estimate of failure rate at start of first interval, α • Estimate of proportionality constant of failure rate over time, β	• Software must be operational • Assumes no new faults are introduced during corrective action • Assumes linear reduction in number of residual faults over time

Table 3.6.2.3-1: Summary of Software Reliability Models (continued)

Model Name	Hazard Function Formula	Required Data or Estimation	Limitations and Constraints
Duane	$\lambda(t) = \frac{\lambda_0 t^b}{t}$	• Time of each failure occurrence, t • Estimate or measurement of initial failure rate, λ_0 • The value of "b" is estimated by: $b = \frac{n}{\sum_{i=1}^{n} \ln(t_n + t_i)}$ from i = 1 to the number of detected failures, n	• Software must be operational
Brooks and Motley (IBM)	Binomial: $P(X = n_i) = \binom{R_i}{n_i} q_i^{n_i} (1 - q_i)^{R_i - n_i}$ Poisson: $P(X = n_i) = \frac{(R_i \phi_i)^{n_i} e^{-R_i \phi_i}}{n_i!}$	• Number of faults remaining at start of i^{th} test, R_i • Total number of faults found in each test, n_i • Test effort required for each effort, K, used in calculation of q_i • Probability of fault detection in the i^{th} test, q_i • Probability of correcting faults without introducing new ones, α, used in calculation of R_i	• Software is developed incrementally • Rate of fault detection is assumed constant over time • Some software modules may have different test effort
Yamada, Ohba & Osaki S-Shape	$\text{Fault Detection Rate} = ab^2 te^{-bt}$	• Time of each failure detection, t • Simultaneous solving of variables a, b	• Software is operational • Fault detection rate is S-shaped over time
Weibull	$MTTF = \frac{b}{a}\Gamma\left(\frac{1}{a}\right)$	• Total number of faults found during each testing interval • The length of each testing interval • Parameter estimation of "a" and "b"	• Failure rate can be increasing, decreasing or constant
Geometric	$D\phi^{t-1}$	• Either time between failure occurrences, or the time of failure occurrence, t • Estimate of constant "D", which decreases in geometric progression as failures are detected: $(0 < \phi < 1)$	• Software is operational • Inherent number of failures assumed to be infinite • Faults are independent and unequal in probability of occurrence and severity
Thompson & Chelson Bayesian	$\frac{f_i + f_0 + 1}{T_1 + T_0}$	• Number of failures detected in each interval, f_i • Length of test time for each interval, T_i	• Corrective action is incorporated into software at end of testing interval • Software is operational • Software is approximately fault free
Rome Laboratory (RL-TR-92-15)	$\lambda(t) = \lambda_0 e^{-\left(\frac{B\lambda_0}{W_0}\right)t}$	• Initial software failure rate, λ_0 • CPU execution time, t, in seconds • RL fault reduction factor, B (default is 0.955) • Initial number of faults per 1000 LOC	

With the number of potential models available, it is not easy to select which model may be most appropriate for a specific situation. Figure 3.6.2.3-1, taken from Reference 2, attempts to provide some guidance on model selection based on the following constraints:

- Failure profiles (failure intensity trend)
- Maturity of software (what phase of its life cycle is the software in)
- Characteristics of software development (how are failure modes detected/mitigated)
- Characteristics of software test
- Existing metrics and data

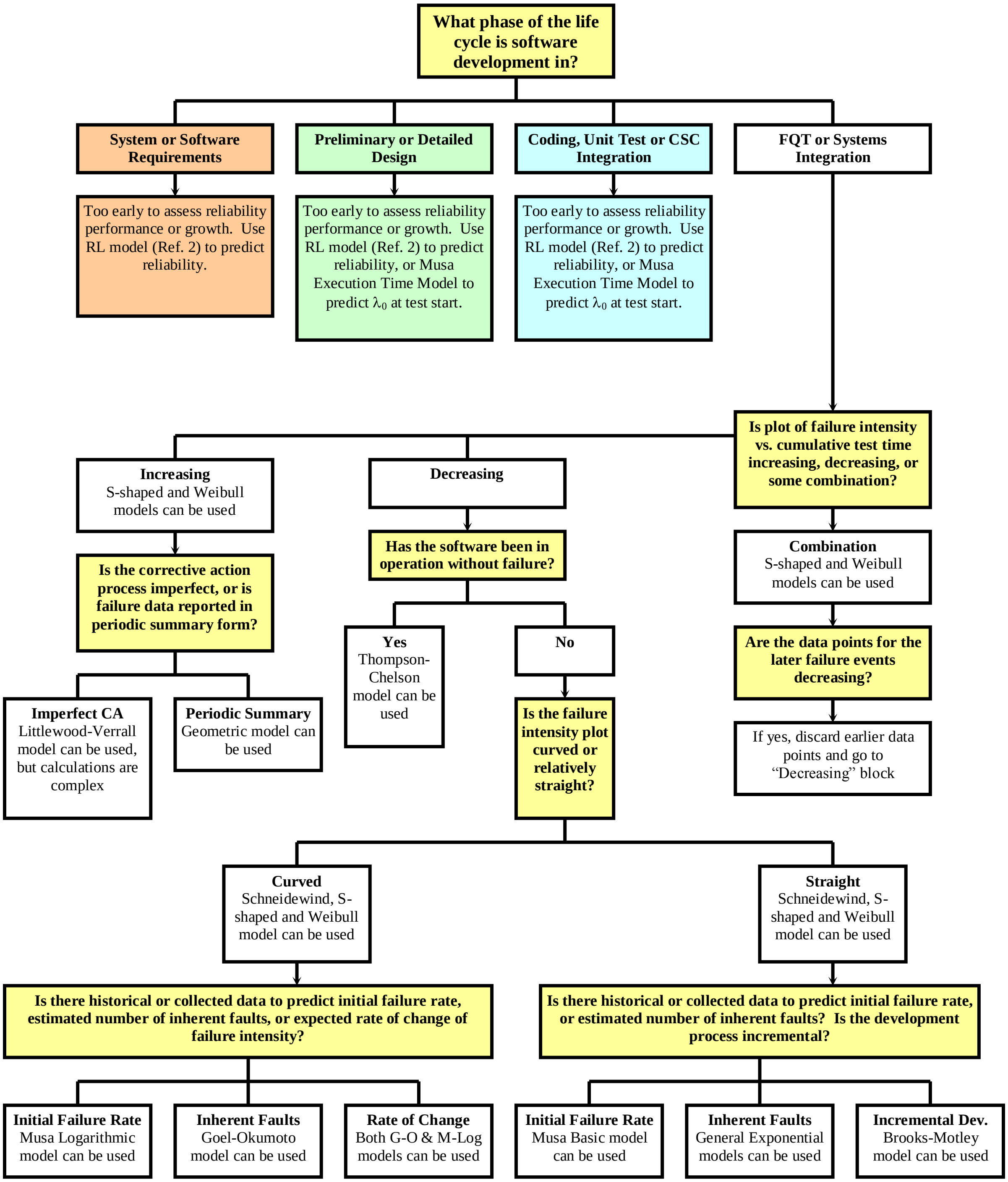


Figure 3.6.2.3-1: Selection of an Appropriate Software Reliability Growth Model

If the plot of failure intensity vs. cumulative test time is showing an increase in failure intensity (negative reliability growth), then you need to make sure that the software is in an operational state, that only unique software failure modes are being counted, and that all time estimates are accurate. If these conditions are satisfied, it is likely that the software is still in the early stages of system development or test.

If the plot of failure intensity vs. cumulative test time is decreasing, you must still make sure that the software is being tested or used in an operational profile that is representative of how it will be used – or misused -- in the field, and that there have been no failures experienced for a reasonably significant period of time.

For More Information:

1. AIAA R-013-1992, "Recommended Practice for Software Reliability", 1993
2. Lakey, P.B. and Neufelder, A.M., "System and Software Reliability Assurance Notebook", Rome Laboratory, RL-TR-97-XX, 1997
3. Musa, J.D., "Software Reliability Engineering: More Reliable Software, Faster Development and Testing", McGraw-Hill, July 1998, ISBN 0079132715

Topic 3.6.2.4: Planning Models Based on AMSAA Projection Methodology (PM2)

As stated in MIL-HDBK-189A:

> "The goal of reliability growth planning is to optimize testing resources, quantify potential risks, and plan for successful achievement of reliability objectives. A well thought out reliability growth plan can serve as a significant management tool in scoping out the required resources to enhance system reliability and improve the likelihood of demonstrating the system reliability requirement. The principal goal of the growth test is to enhance reliability by the iterative process of surfacing failure modes, analyzing them, implementing corrective actions (fixes), and testing the "improved" configuration to verify fixes and continue the growth process by surfacing remaining failure modes. A critical aspect underlying this process is ensuring that there are adequate resources available to support the desired growth path. This includes addressing program schedules, amount of testing, resources available, and the realism of the test program in achieving its requirements. Planning activities include establishing test schedules, determining resource availability in terms of facilities and test equipment, and identifying test personnel, data collectors, analysts and engineers. Another factor necessary for a successful growth program is allowing for sufficient calendar time during the program to analyze, gain approval and implement corrective actions. Planning is quantified and reflected through a reliability growth program plan curve. This curve may be used to establish interim reliability goals throughout the test program. Two significant benefits of reliability growth planning are:
>
> a. Can perform trade-offs with test time, initial reliability, final reliability, confidence levels, requirements, etc., to develop a viable test program.
>
> b. Can assess the feasibility of achieving a requirement given schedule and resource constraints by using historical values for parameters (e.g., growth rate)."

Continuous PM2

To mature the reliability of a complex system under development, it is important to formulate a detailed reliability growth plan. One aspect of this plan is a depiction of how the system's reliability is expected to increase over the developmental test period. The depicted growth path serves as a baseline against which reliability assessments can be compared. Baseline planning curves for Department of Defense (DoD) systems have frequently been developed in the past utilizing the assumed reliability growth pattern specified in the original MIL-HDBK-189 document (1981). This growth relationship is between the reliability, expressed as the mean test duration[3] between system failures, and a continuous measure of test duration such as time or mileage. The equation governing this growth pattern was motivated by the empirically-derived linear relationship observed for a number of data sets by Duane between the developmental system cumulative failure rate and the cumulative test time when plotted on a log-log scale.

MIL-HDBK-189A, Section 5.5, discusses and derives a non-empirical relationship between the system MTBF and cumulative test time that can be utilized for reliability growth planning. This relationship is derived from a fundamental relationship between the expected number of failure modes surfaced and the cumulative test time. The functional form of this fundamental relationship is well known and is easily established. The PM2 methodology develops an approximation to this relationship that is suitable for reliability growth planning. One significant advantage to the PM2 approach is that it does not rely on an empirically-derived relationship such as the Duane-based approach. The MIL-HDBK shows how the cumulative relationship between the expected number of discovered failure modes and the test time naturally gives rise to a reliability growth relationship between the expected system failure intensity and the cumulative test time. The PM2 approximation for the resulting growth pattern avoids a number of deficiencies associated with the Duane/MIL-HDBK-189 approach to reliability growth planning.

[3] For convenience, subsequent discussion will use time as the basis for test duration, although test duration can also be based on miles, cycles, operations, etc..

Section 5.5.3 of MIL-HDBK-189A develops the exact expected system failure intensity and parsimonious approximations suitable for reliability growth planning. These functions of test time are derived from the exact and planning approximation relationships between the expected number of surfaced failure modes and the cumulative test time. The exact relationship is expressed in terms of the number of potential failure modes, k, and the individual initial failure mode rates of occurrence. Parsimonious approximations to this relationship are obtained. The first approximation utilizes the number of potential failure modes and several additional parameters. The second approximation addressed is the limiting form of the first approximation as the number of potential failure modes increases. This approximation is suitable for complex systems or subsystems. The approximations are derived through consideration of an MTBF projection equation. This equation arises from considering the problem of estimating the system MTBF at the start of a new test phase after implementing corrective actions to failure modes surfaced in a preceding test phase.

MIL-HDBK-189A, Section 5.5.4, contains simulation results. The simulations are conducted to obtain actual patterns for the cumulative number of surfaced failure modes versus test time for random draws of initial mode failure rates from several parent populations, and for a geometric sequence of initial mode failure rates. The resulting stochastic realizations are compared to the theoretical expected number of potential surfaced failures modes and to the parsimonious approximations. Random draws for failure mode FEFs are used to simulate corrective actions to discovered failure modes. Using the simulated corrective actions, the relationship between the expected system failure intensity and cumulative test time is simulated for various sets of mode initial failure rates. This relationship is obtained under the assumption that the system failure intensity associated with a cumulative test time, t, reflects implementation of corrective actions to the modes surfaced by time "t" with the associated randomly drawn FEFs. The resulting system MTBF versus test time relationship is compared to the corresponding relationship established for planning purposes.

MIL-HDBK-189A, Section 5.5.5, derives expressions for a reliability projection scale parameter that is utilized in the parsimonious approximations. The projection parameter is expressed in terms of basic planning parameters. The resulting MTBF approximations are compared to the reciprocals of the exact expected system failure intensity and stochastic realizations of the system failure intensity, and to MIL-HDBK-189 MTBF approximations based on planning parameters. The comparisons are done for several reliability growth patterns.

Section 5.5.6 of MIL-HDBK-189A addresses the relationship between the theoretical upper bound on the achievable system MTBF, termed the growth potential, and the planning parameters. The projection scale parameter discussed in Section 5.5.5 of the MIL-HDBK is then expressed in terms of planning parameters and the MTBF growth potential. It is shown that the scale parameter becomes unrealistically large if the goal MTBF is chosen too close to the growth potential, or if the allocated test time to grow from the initial to goal MTBF is inadequate.

Finally, Section 5.5.7 of the MIL-HDBK indicates how to construct a sequence of MTBF target values that start at an expected or measured initial MTBF and end at the goal MTBF. It is shown that the parsimonious approximation to the reciprocal of the expected system failure intensity can be used for this purpose in conjunction with a test schedule that specifies the expected monthly hours to be accumulated on the units under test, and the planned corrective action periods.

Table 3.6.2.4-1 highlights the options, required inputs and calculated outputs associated with the PM2-Continuous Reliability Growth Planning Model (commonly referred to as just "PM2").

Table 3.6.2.4-1: PM2-Continuous Reliability Growth Planning Model Options, Required Inputs and Calculated Outputs

Model	Options	Required Inputs	Calculated Outputs

PM2 (Continuous)	• Construct a reliability growth planning curve for continuous systems	• Choose “General” or “Detailed” Schedule Input Option
		General Schedule Inputs (for each Test Phase):
		• Test Phase Name
		• Mission Time in Test Phase
		• Corrective Action Period (CAP) at End of Phase (Yes/No)
		• Corrective Action Lag Time for Individual CAP
		Detailed Schedule Inputs (for each Test Phase):
		• Test Phase Name
		• Period Length
		• Test Phase Length (in periods)
		• Number of Items Used in Test Period
		• Corrective Action Lag Time (in periods)
		• **For Each Test Item in Test Phase**
		  ○ Planned Number of Test Hours in each Period
		• Choose “IOT Incorporated in Planning Curve?” (Yes/No)
		IOT is **Not** Incorporated in Planning Curve:
		• Requirement MTBF
		• Initial MTBF
		• Management Strategy
		• Average FEF
		IOT **is** Incorporated in Planning Curve:
		• Same as if it is not, plus:
		  ○ IOT Training Test Time
		  ○ IOT Phase Test Time
		  ○ Assumed DT-to-IOT Degradation Factor
		  ○ Confidence Level for IOT LCB
		  ○ Probability of Acceptance in IOT using LCB
		  ○ Test Phase for ASA(ALT) – (N/A, 1st or 2nd)
		  ○ **For IOT OC Analysis:**
		    ▪ Confidence Level for LCB
		    ▪ Probability of Acceptance at LCB
			• Probability of Acceptance in IOT using Point Estimate
			• Goal MTBF in IOT
			• Goal MTBF in DT
			• Growth Potential
			• Ratio of Goal MTBF in DT to Growth Potential
			For ASA(ALT) Threshold:
			• ASA(ALT) Threshold
			• Test Length
			• Maximum Number of Failures
			• LCB for ASA(ALT) Threshold
			• Probability of Acceptance using LCB
			• Probability of Acceptance using Point Estimate
			For IOT OC Analysis:
			• Maximum Number of Failures
			• LCB for Requirement
			• Goal MTBF in IOT
			• Ratio of Goal MTBF in DT to Growth Potential
			• Probability of Acceptance using LCB
			• Probability of Acceptance using Point Estimate
			• Expected Number of B-modes by Time, t
			• Expected Rate of Occurrence of B-modes by Time, t
			• Percent of Initial B-mode Failure Intensity Surfaced by Time, t
			• Expected Number of Failures (All or B-Mode only)

Example

Due to the complexity and depth of the calculations, MIL-HDBK-189A does not provide a detailed example of PM2 functionality. Section 5.5.8 of the MIL-HDBK does provide a high-level description for generating a planned reliability growth curve path.

Suppose a test schedule is laid out that defines a planned number of miles accumulated on the units under test per month. Also, suppose that the test schedule specifies blocks of calendar time for implementing corrective actions. Finally, for planning purposes, assume that in order for a failure mode to be addressed during an upcoming corrective action period, it must occur four months prior to the start of the test period. For this situation, the MTBF could be represented by a constant value between the ends of corrective action periods and between the start of testing and the end of the first scheduled corrective action period (CAP). For such a test plan, jumps in MTBF would be portrayed at the conclusion of each CAP.

Figure 3.6.2.4-1 depicts a detailed PM2 reliability growth planning curve for a complex system for the case where A-mode and B-mode failure categories are defined.

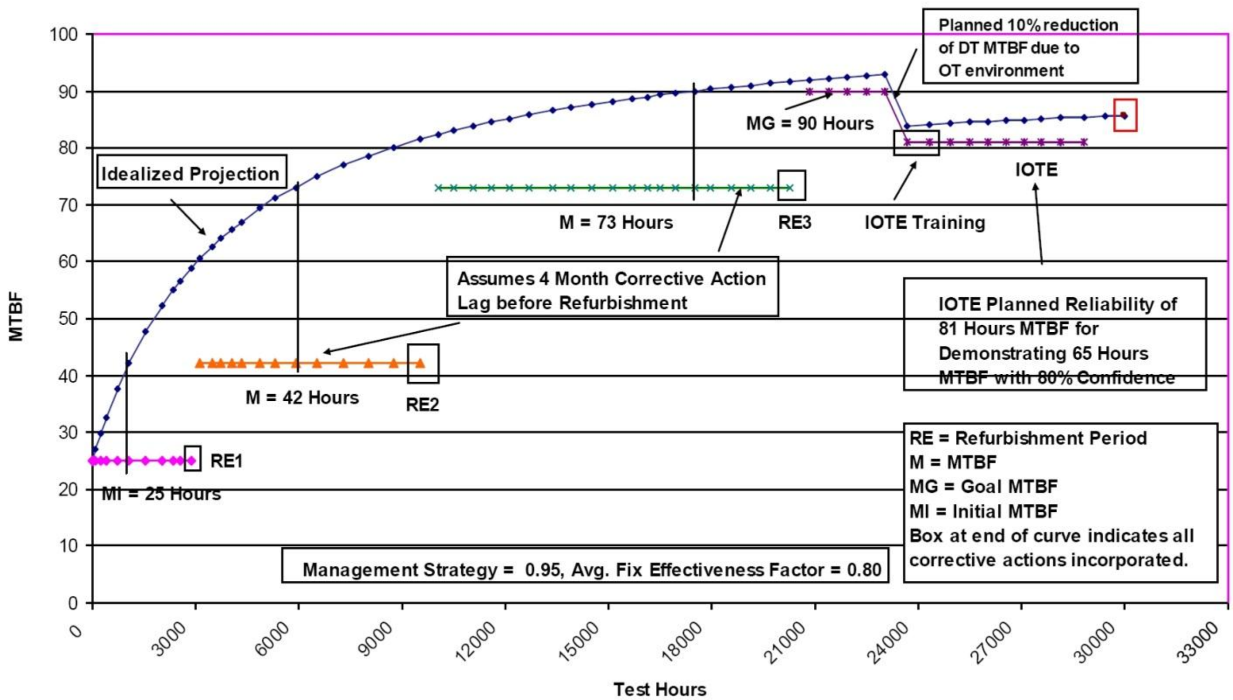


Figure 3.6.2.4-1: PM2-Continuous Reliability Growth Planning Curve

The “blue” continuous curve represents a plot of the instantaneous MTBF over time, t, given by the equation:

$$M_{PL}(t) = \{\rho_{PL}(t)\}^{-1} = \frac{1}{\lambda_A + (1-\mu_d^{'})(\lambda_B - h_B(t)) + h_B(t)}$$

where λ_A is the failure intensity due to A-modes, λ_B is the initial failure intensity due to B-modes (thus, $\lambda = \lambda_A + \lambda_B$), $h_B(t)$ is the expected failure intensity due to the set of B-modes not discovered by time “t”, and $\mu_d^{'}$ is the average FEF that would be realized for the B-modes if all were discovered during test. The scale parameter, β, is calculated from the PM2 planning parameter inputs:

$$\beta = \left(\frac{1}{T}\right)\left(\frac{1-\frac{M_I}{M_G}}{\left(MS \bullet \mu_d^{'}\right)-\left(1-\frac{M_I}{M_G}\right)}\right)$$

where MS = λ_B/λ. The planning parameter, MS, is the management strategy discussed throughout this book, representing the fraction of the total system failure intensity, λ, that is due to the initial B-mode failure intensity. If there were no A-modes defined for the system, then the most aggressive MS would presumably be 1.0.

Note that the value of MTBF at time, t, is the system MTBF one plans to attain after all corrective actions to B-modes discovered (seen) during the test period are implemented. The MTBF steps are constructed from the continuous "blue" curve, the schedule of CAPs, and the assumed average corrective action implementation lag. From Figure 6.4.1-1, note that the goal MTBF, M_G, of 90 hours was chosen to be larger than the required MTBF, M_R, of 65 hours, which is the MTBF to be demonstrated during a follow-on Initial Operational Test & Evaluation (IOT&E). The IOT&E is an operational demonstration test of the system's suitability for fielding. In such a test it may be required to demonstrate, with a measure of statistical confidence, that a pre-defined MTBF goal has been achieved. For this example, the measure of assurance is a demonstration of M_R at the 80% statistical confidence level. In order to have a reasonable probability of demonstrating this value, the system must enter the IOT&E with an MTBF value that is greater than the required value. This needed value can be determined by a well-known statistical procedure (MIL-HDBK-781) based on the IOT&E test length, the desired confidence level of the statistical demonstration, and the specified probability of being able to achieve the statistical demonstration. After determining this MTBF value, one can determine what the goal MTBF, M_G, should be at the conclusion of the development test. The value of M_G should be the goal MTBF to be achieved just prior to the IOT&E training period that precedes the actual IOT&E. The goal MTBF associated with the development test environment must be chosen sufficiently above the IOT&E entrance value MTBF so that the operational test environment does not cause the reliability of the test units to fall below the entrance value during the IOT&E. The significant drop in MTBF often seen during IOT&E tests could be attributable to operational failure modes that were not discovered during the developmental test. In the example of Figure 6.4.1-1, a derating factor of 10% was used to obtain the MTBF goal, M_G, from the IOT&E entrance MTBF value.

Figure 3.6.2.4-2, taken from MIL-HDBK-189C, illustrates the growth planning curve as a function of calendar time and the step function growth pattern as corrective actions are incorporated at planned times during the test program. The depiction of growth in an Idealized Growth Curve does not preclude the possibility that some fixes may be implemented outside of corrective action periods, i.e., during a test phase. These would typically be fixes to maintenance or operational procedures. They could also include easily diagnosed and implemented design changes to hardware or software. However, any significant reliability growth would typically be expected to occur due to groups of fixes that are scheduled for implementation in CAPs. These would include fixes whose implementation would involve intrusive physical procedures. If fixes are expected to be applied during a test phase, then a portion of the jump in MTBF (or drop in system failure intensity) portrayed at the conclusion of a test phase CAP would be realized during the test phase prior to the associated CAP. Thus, a test phase step in an Idealized Growth Curve simply portrays the test phase MTBF that would be expected under the plan if no fixes were implemented during that test phase.

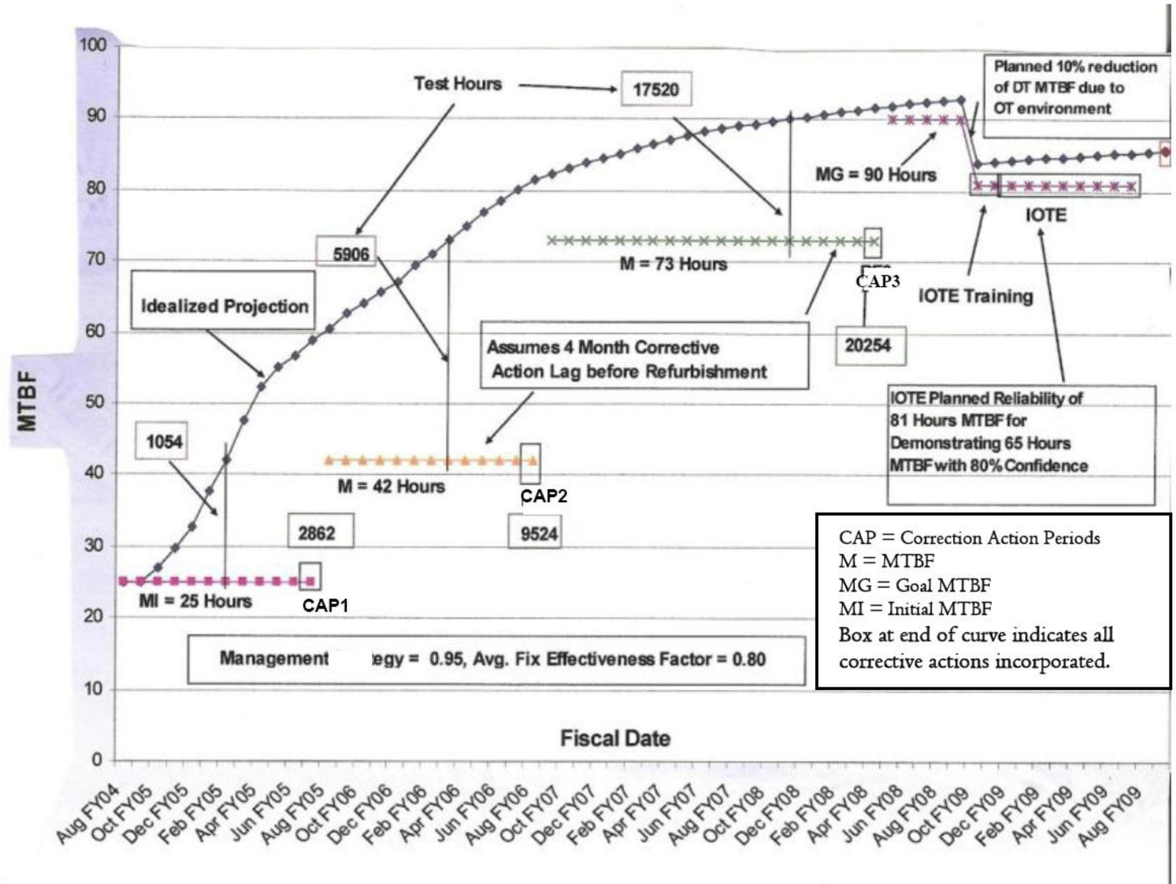


Figure 3.6.2.4-2: PM2-Continuous Reliability Growth Planning Curve in Calendar Time (taken from MIL-HDBK-189C, Figure 28, Best Available Image)

Discrete PM2

According to MIL-HDBK-189A, Section 5.6, the mathematical developments for PM2-Discrete represent the first reliability growth planning methodology developed specifically for discrete systems. Thus, it represents the first quantitative method that reliability practitioners and program managers can use for formulating detailed reliability growth plans in the discrete-usage domain. The PM2-Discrete approach is not just a reliability growth planning model. It is a robust reliability growth planning methodology that possesses concurrent measures of programmatic risk and system maturity. For instance, PM2-Discrete offers several reliability growth management metrics of fundamental interest that practitioners may use when assessing the ability of a proposed T&E plan to achieve the desired result. These metrics include:

- Expected number of failures observed by trial, t
- Expected number of failure modes observed by trial, t
- Expected reliability on trial, t, under failure mode mitigation
- Expected reliability growth potential[4]
- Expected probability of failure on trial, t, due to a new failure mode
- Expected fraction surfaced of the system probability of failure on trial, t

[4] The ***reliability growth potential*** is the theoretical upper limit on reliability that can be achieved by finding and fixing all B-modes with a specified level of fix effectiveness.

The PM2-Discrete equations associated with these metrics, as well as the required inputs, are summarized in Table 3.6.2.4-2 and discussed throughout this section.

Table 3.6.2.4-2: PM2-Discrete Reliability Growth Planning Model Options, Required Inputs and Calculated Outputs

Model	Options	Required Inputs	Calculated Outputs
PM2-Discrete[5]	• Construct a reliability growth planning curve for discrete systems	• Total Number of Trials (T) • Management Strategy (MS) • Initial System Reliability (R_I) • Planned Average FEF (μ) • Reliability Goal for the System (R_G) • Total Number of Trials to Lag Time Before the Last Corrective Action Phase (T_L) • Total Number of Unique B-Modes <u>For Each Unique B-Mode:</u> • Achieved FEF	• Number of Failures Observed by Trial, t • Number of Failure Modes Observed by Trial, t • Portion of System Reliability Comprised of A-Modes • Portion of System Reliability Comprised of B-Modes • Reliability on Trial, t, under Instantaneous Failure Mode Mitigation • Reliability Growth Potential • Probability of Failure on Trial, t, due to a New B-Failure Mode • Fraction Surfaced of the Initial System Probability of Failure due to B-Modes Through Trial, t • Estimated Reliability Growth Parameter • Estimated Reliability Growth Potential • Estimated Model Scale Parameter

The PM2-Discrete methodology presented in MIL-HDBK-189A consists of deriving several model equations of relevant interest. These model equations constitute the analytical framework from which a number of different reliability growth management metrics may be estimated. The PM2-Discrete metrics include:

- Expected Reliability (Idealized Planning Curve):

$$R(t) = R_A \bullet R_B^{\,1-\mu\left(t-1/_{n+t-1}\right)}$$

where R_A is the fraction of system reliability comprised of failure modes that **<u>will not</u>** be addressed/mitigated through corrective action (A-modes); R_B is the fraction of system reliability comprised

[5] The material presented for the PM2-Discrete model is derived from Draft MIL-HDBK-189C, dated 17 May 2010 and, therefore, subject to change. As of 31 March 2011, AMSAA has indicated that their PM2-Discrete Model software tool will not be released until validation of the tool has been completed.

of failure modes that **will** be addressed/mitigated through corrective action (B-modes); μ is the planned FEF; and “n” is the shape parameter of the beta distribution that represents pseudo trials.

The equations required to calculate R_A, R_B and “n” are:

$$R_B = R_I^{MS}$$

$$R_A = R_I^{(1-MS)}$$

$$n = (T_L - 1) \bullet \frac{\ln(R_{GP}/R_G)}{\ln(R_G/R_I)}$$

- Reliability Growth Potential:

$$R_{GP} = R_I^{1-MS \bullet \mu}$$

- Expected Number of Failures:

$$f(T) = \sum_{j=1}^{r} \ln R_j^{-T_j}$$

where “r” is the number of test phases corresponding to the fixed configurations of the system with respect to reliability, and the individual summation terms are interpreted as the expected number of failures in test phase, j.

- Expected Number of Failure Modes On/Before Trial, t:

$$\mu(T) = \sum_{j=0}^{t-1} \frac{\ln R_I^{-n}}{n+j}$$

- Expected Probability of Failure Due to a New Mode on Trial, t:

$$h(t) \equiv 1 - R_A \bullet R_B^{\left(n/_{n+t-1}\right)}$$

- Expected Fraction Surfaced of System Probability of Failure

$$\phi(t) \equiv \frac{1 - R_A \bullet R_B^{\left(n/_{n+t-1}\right)}}{1 - R_I}$$

MIL-HDBK-189A does not provide an example of PM2-Discrete functionality. Section 5.6 of the MIL-HDBK does provide detailed discussion and derivations of the relevant equations just discussed.

For More Information:

1. Nicholls, D., P. Lein, T. McGibbon, “Achieving System Reliability Growth Through Robust Design and Test”, Reliability Information Analysis Center, 2011.
2. MIL-HDBK-189, “Reliability Growth Management”, 13 February 1981 (Revision currently being developed).

Topic 3.6.3: Reliability Demonstration/Qualification Testing

Reliability demonstration/qualification testing (RDT/RQT) is conducted as part of the system test and evaluation process. The typical objective of RDT/RQT is to determine if the system under test meets the specified MTBF requirements. To accomplish this, the system is operated in a specified manner for a designated time period and failures are recorded and evaluated as the test progresses. Acceptance of the system is based on the system demonstrating a minimum acceptable reliability. There are a number of test methods and statistical procedures designed to measure and validate system reliability, most of which assume the applicability of the exponential distribution.

1. **Reliability Sequential Testing**. The purpose of RDT/RQT is to provide evaluation of developmental progress, as well as the assurance that specified requirements have been met prior to proceeding to the Production and Deployment Phase of the life cycle. The system under test is operated in a manner that reflects the mission cycles in a realistic operational environment (see Figure 3.6.3-1). During RDT/RQT, there are three possible decisions: (1) accept the system, (2) reject the system, or (3) continue to test.

   Figure 3.6.3-2 represents actual experience in testing a hypothetical system. Referring to the figure, the specified MTBF for the system is 400 hours, and the maximum designated test time used for the sequential test plan is 4,000 hours (multiple of ten times the specified MTBF). The test approach involves the selection of a designated quantity of systems (equipments), operating the system under prescribed performance conditions over an extended period of time and monitoring the system for failure. As failures (events) occur, appropriate corrective maintenance actions are determined and the system is repaired, after which it is returned to test. Failure analysis of each event should be performed down to its root cause. Trends may be established if more than one failure is traceable to the same failure mode (pattern failures). In such cases, an engineering design change may be initiated to preclude the recurrence of failures of the same type.

2. **Reliability Acceptance Testing.** Production Reliability Acceptance Testing (PRAT) may be performed during full-scale production on a 100% or a sampling basis. To determine the effects of the production process on system reliability, it may be feasible to select a sample number of equipments from each production lot and test them in the same manner as for RDT/RQT. The sample may be based on a percentage of the total equipments spread over the entire production period, or a set number of equipment(s) selected during a specified calendar time period (e.g., three items of equipment per month throughout the production phase). The selected equipment is tested and an assessed MTBF is derived either from the test data. This value is compared against the specified MTBF and the measured value determined from earlier qualification testing. Positive or negative MTBF trends may be determined by plotting the resultant values as testing progresses.

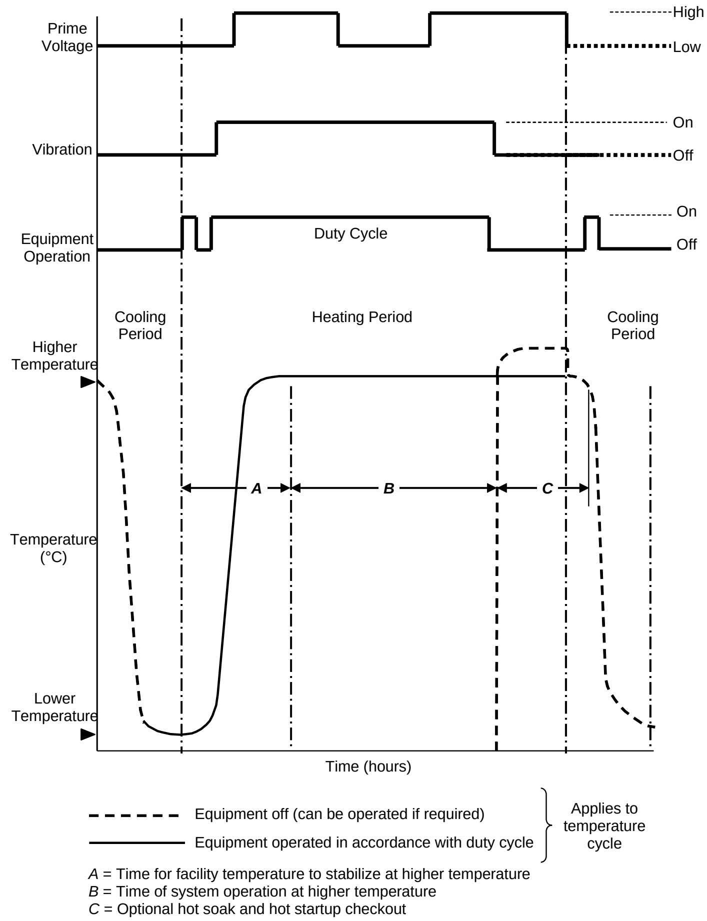


Figure 3.6.3-1: Sample Environmental Test Cycle

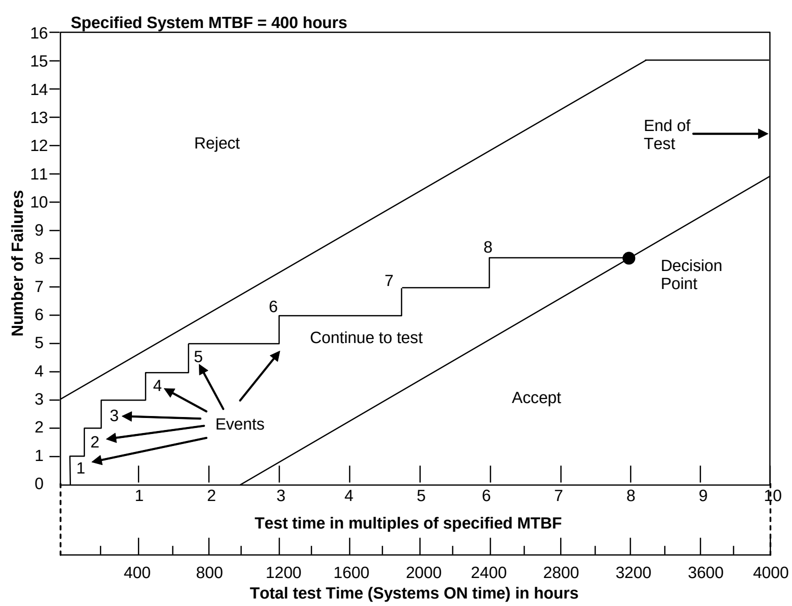


Figure 3.6.3-2: Hypothetical RDT/RQT Test Plan and Results

3. **Reliability Life Testing.** The two basic forms of life testing are:

   a. Life tests based on a fixed-test time.
   b. Life tests based on the occurrence of a predetermined number of failures.

   In the first approach, a fixed test time is computed and a specified number of failures is predetermined. The system is accepted if the actual number of failures at the end of the scheduled test time is equal to or less than the predetermined quantity of failures. In the second approach, a test plan is developed that specifies a predetermined number of failures and a computed test time based on an expected system failure rate. Testing continues until the specified quantity of failures occurs. The system is accepted if the test time is equal to or greater than the computed time at the point

In addition to the statistical basis to RDT/RQT, other important test considerations include those shown in Table 3.6.3-1.

Table 3.6.3-1: Considerations for RDT/RQT

Consideration	Comments
Definition of failure	Before any testing begins, agreement is needed with the customer as to what constitutes a failure. Ideally, this should have already been defined by the failure definitions and scoring criteria contained in the contractual specification. Do transient/intermittent events represent a failure? Is degraded performance considered a failure and, if so, how much degradation is acceptable, i.e., what is the threshold level signifying failure? What actions and resolution are to take place for each experienced failure?
Test environment	The ideal environmental conditions and operating profile will represent what the system will experience in its intended use environment.
System configuration	Is the item under test representative of the hardware/software configuration that will be used by the customer in the field, and is it being exercised in a similar manner?
Test monitoring	The system should be monitored for correct performance at reasonable time intervals using techniques (preferably automated) that will all capture failure events, including intermittencies
Failure analysis	Will all failure modes be analyzed for root cause and appropriate corrective action that will be verified for success? If not all, then which ones (e.g., safety-critical, mission-critical, reliability-critical, etc.)
Special conditions	While the number of failures may be acceptable, attention should be paid to any pattern of failures that may occur, as trends may indicate an opportunity for correction. Ideally, a corrective action should be identified for any experienced failure mode, even if the number of failures is considered acceptable.

Table 3.6.3-2 summarizes the important definitions that relate directly to RDT/RQT.

Tables 3.6.3-3, 3.6.3-4 and 3.6.3-5 provide an overview of three basic types of RDT/RQT:

- Failure-free execution interval test
- Fixed-length test
- Probability-ratio sequential test (PRST)

Figure 3.6.3-3 provides a conceptual description for developing a RDT/RQT that is based on satisfactory levels of both producer and consumer risk when testing is to be performed. Once these risk values are defined, the corresponding values of “n” (the number of allowed failures) and “t” (the sum of the required test times) can be calculated.

Table 3.6.3-2: Definitions Related to RDT/RQT

Term	Definition
True Failure Rate (λ) or True MTBF (θ)	Represents the actual, unknown failure rate (λ) or mean time between failure (θ) of the system. Remember that MTBF = $1/\lambda$
Lower Test Failure Rate (λ_1) or Lower Test MTBF (θ_1)	The lower test failure rate and lower test MTBF represent those values of λ or θ which are considered unacceptable to the customer, and will result in a high probability of system **rejection**
Upper Test Failure Rate (λ_0) or Upper Test MTBF (θ_0)	The upper test failure rate and upper test MTBF represent those values of λ or θ which are considered acceptable to the customer, and will result in a high probability of system **acceptance**
Discrimination Ratio (d or δ)	Represents a reliability demonstration test plan parameter which is a measure of the power of the test to reach an accept/reject decision quickly. In general, the higher the discrimination ratio, the shorter the test to prove statistical significance. **Failure rate discrimination ratio:** $\mathbf{d} = \frac{\boldsymbol{\lambda}_1}{\boldsymbol{\lambda}_0}$ **MTBF discrimination ratio:** $\mathbf{d} = \frac{\mathbf{MTBF}_0}{\mathbf{MTBF}_1}$
Producer's or Supplier's Risk (α)	The probability of rejecting equipment with a true failure rate or true MTBF equal to the upper test failure rate (λ_0) or MTBF (θ_0), i.e., the probability of rejecting good systems
Consumer's Risk (β)	The probability of accepting equipment with a true failure rate or true MTBF equal to the lower test failure rate (λ_1) or MTBF (θ_1), i.e., the probability of accepting bad systems

Table 3.6.3-3: Failure-Free Execution Interval Test

Description
A failure-free execution interval test requires that a given number of samples be tested for a specified time. If no failures occur during that test, the system is considered as having met its reliability requirements. The determination of sample size and test length is accomplished by considering the system reliability function. This test will accept software with an MTBF higher than θ_0 (lower than λ_0) more quickly than a fixed duration test. The appropriate formulae for the exponential distribution are: $\text{Consumer Risk} = \mathbf{e}^{-\mathbf{n}(\lambda_1 \mathbf{t})}$ based on failure rate, or $\text{Consumer Risk} = \mathbf{e}^{-\mathbf{n}\left(\mathbf{t}/\theta_1\right)}$ based on MTBF where, t = the amount of time to test with no failures experienced n = number of "samples" being tested **<u>Example</u>:** If the consumer is willing to accept a 20% risk (β) of accepting "bad" products (unacceptable MTBF = θ_1), and 50 items are to be subjected to test, the total test time with zero failures required to statistically prove that the software or system is acceptable is: $0.20 = \mathbf{e}^{-50\left(\mathbf{t}/\theta_1\right)}$ $\ln(0.20) = -50\left(\mathbf{t}/\boldsymbol{\theta}_1\right)$ $-1.609/-50 = \frac{\mathbf{t}}{\boldsymbol{\theta}_1}$ $\mathbf{t} = 0.032(\boldsymbol{\theta}_1)$

Table 3.6.3-4: Fixed-Length RDT/RQT

Description
A fixed-length RDT/RQT is used when the amount of test time (and its associated costs) must be known in advance. This type of test provides a demonstrated MTBF (or failure rate) to a desired confidence level, as well as providing criteria to reach accept/reject decisions for the test (based on the number of failures experienced during the test). Based on the exponential distribution, and letting "n" be the maximum number of failures allowed during the test, the equations for the "bad" failure rate (λ_1) and MTBF (θ_1), i.e., Consumer's Risk, are given as: $$\mathbf{P}\{n\} = \text{Consumer's Risk} = \sum_{0}^{\mathbf{n}} \frac{\left(\mathbf{t}/\boldsymbol{\theta}_1\right)^{\mathbf{n}} \mathbf{e}^{-\left(t/\theta_1\right)}}{\mathbf{n}!} \quad \text{for MTBF}$$ $$\mathbf{P}\{n\} = \text{Consumer's Risk} = \sum_{0}^{\mathbf{n}} \frac{\left(\mathbf{t}\boldsymbol{\lambda}_1\right)^{\mathbf{n}} \mathbf{e}^{-\left(t\lambda_1\right)}}{\mathbf{n}!} \quad \text{for Failure Rate}$$ The equations for the "good" failure rate (λ_0) and MTBF (θ_0), i.e., Producer's Risk, are given as: $$\mathbf{P}\{\mathbf{n}\} = \text{Producer's Risk} = 1 - \sum_{0}^{\mathbf{n}} \frac{\left(t/\boldsymbol{\theta}_0\right)^{\mathbf{n}} \mathbf{e}^{-\left(t/\theta_0\right)}}{\mathbf{n}!} \quad \text{for MTBF}$$ $$\mathbf{P}\{\mathbf{n}\} = \text{Producer's Risk} = 1 - \sum_{0}^{\mathbf{n}} \frac{\left(t\boldsymbol{\lambda}_0\right)^{\mathbf{n}} \mathbf{e}^{-\left(t\lambda_0\right)}}{\mathbf{n}!} \quad \text{for Failure Rate}$$ In order to formulate a test that is based on satisfactory attainment of both consumer's and producer's risk, the value for both need to be defined, and then their corresponding equations solved for values of "n" and "t" that will simultaneously satisfy both risk equations.

Table 3.6.3-5: Probability-Ratio Sequential RDT/RQT

Description
A sequential RDT/RQT will accept a system that has a failure rate much lower than λ_0 (MTBF much higher than θ_0) and reject a system that has a failure rate much higher than λ_1 (MTBF much lower than θ_1) more quickly than a fixed-length test that has similar Consumer Risk, Producer Risk and Discrimination Ratio parameters. The expected test time may be significantly longer, however, as it assumes that the true failure rate (or MTBF) is equal to the upper test failure rate (or MTBF), rather than the mean. The PRST is based on the ratio of two probabilities (from Reference 1): 1. The probability that a combination of failures and test time will occur when the test items are based on the "lower" failure rate or MTBF 2. The probability that a combination of failures and test time will occur when the test items are based on the "upper" failure rate or MTBF If the first probability is sufficiently higher than the second, then a reject decision can be made. If the opposite is true, then an accept decision can be made. If the ratio of the probabilities is not sufficient to warrant an accept or reject decision, testing continues to an arbitrarily determined decision point to ensure that time and money are not unduly "wasted". The boundaries for any such chart can be generated using the following equations: Define: $\mathbf{A} = \ln \frac{\text{Consumer's Risk}}{(1\text{- Producer's Risk})} = \ln \frac{\boldsymbol{\beta}}{(1-\boldsymbol{\alpha})}$ $\quad \mathbf{B} = \ln \frac{(1\text{- Consumer's Risk})}{\text{Producer's Risk}} = \ln \frac{1-\boldsymbol{\beta}}{\boldsymbol{\alpha}}$ Given that the definition of the discrimination ratio is "d" and that "n" is the failure number, the boundary between the "Reject" and "Continue" regions of the chart is given by the equation: $$\mathbf{Boundary}_{\mathbf{R-C}} = \frac{\mathbf{A} - \mathbf{n} * \ln(\mathbf{d})}{1-\mathbf{d}}$$ and the boundary between the "Continue" and "Accept" regions of the chart is given as: $$\mathbf{Boundary}_{\mathbf{C-A}} = \frac{\mathbf{B} - \mathbf{n} * \ln(\mathbf{d})}{1-\mathbf{d}}$$

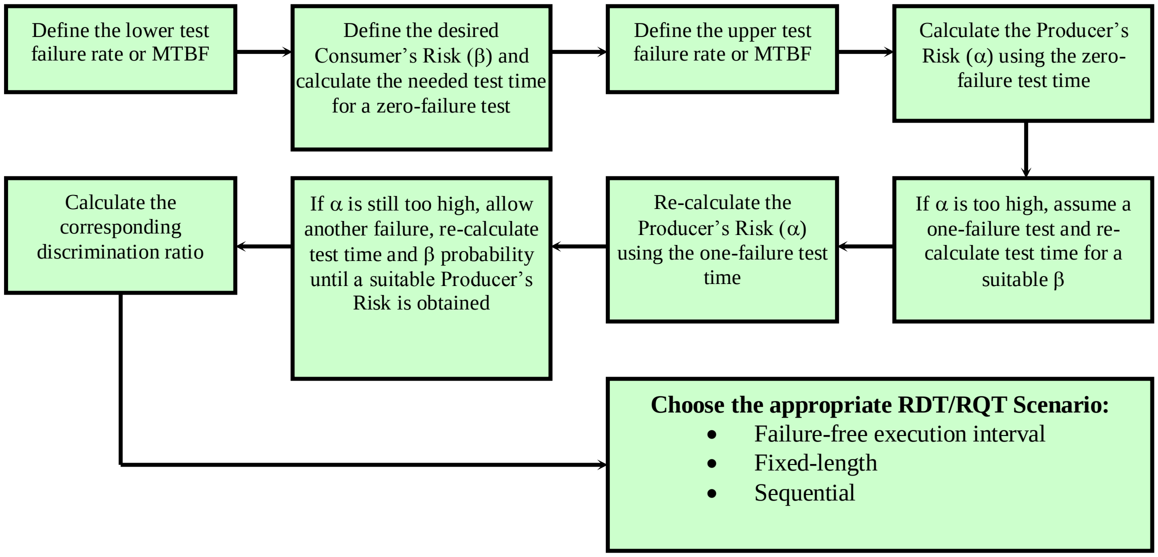


Figure 3.6.3-3: Conceptual Overview for Defining a RDT/RQT

Tables 3.6.3-6, 3.6.3-7, 3.6.3-8 and 3.6.3-9 represent abbreviated versions of tables found in the literature or available in MIL-HDBK-781. Figure 3.6.3-2 provides a graphical representation of what a typical sequential test graph looks like. Table 3.6.3-10 provides factors for calculation of MTBF confidence intervals around reliability demonstration test data.

RDT/RQT and the Poisson Distribution

The Poisson distribution is useful in calculating the probability that a certain number of failures will occur over a certain length of time for systems exhibiting exponential failure distributions (e.g., non-redundant complex systems).

Example 1: If the true MTBF of a system is 200 hours and a reliability demonstration test is conducted for 1000 hours, what is the probability of accepting the system if three or less failures are allowed?

Solution: Expected number of failures = $\lambda t = \frac{t}{MTBF} = \frac{1000}{200} = 5$

From Table 3.6.3-11, the probability of three or less failures (probability of acceptance) given that five are expected is 0.265. Therefore, there is only a 26.5 percent chance that this system will be accepted if subjected to this test.

Example 2: A system has an MTBF of 50 hours. What is the probability of two or more failures during a 10-hour field reliability demonstration test?

Solution: Expected number of failures = $\frac{t}{MTBF} = \frac{10}{50} = 0.20$

The probability of two or more failures is one minus the probability of one or less failures.

From Table 3.6.3-12, $P(r \leq 1)$ when .2 are expected is 0.982.

$P(r \geq 2) = 1 - P(r \leq 1)$
$1 - .982 = .018$

Therefore, there is a very remote chance (1.8 percent) that a system with a 50-hour MTBF will experience two or more failures during a 10-hour test.

Example 3: A system has an MTBF of 50 hours. What is the probability of experiencing exactly two failures during a 10-hour field reliability demonstration test?

Solution: Expected number of failures = $\frac{t}{MTBF} = \frac{10}{50} = 0.20$

From Table 7.5.2-11, the probability of experiencing exactly two failures when 0.20 are expected is 0.017 or 1.7 percent. It should be noted that the probability of experiencing two or more failures, as determined in the last example, can also be determined from this table by adding P(r = 2) + P(r = 3) when .2 are expected.

Table 3.6.3-6: Sequential Test Plan for 10% Risks and Discrimination Ratio = 2.0

Number of Failures	Rejection ($t \leq \theta_1$*Table Entry)	Acceptance ($t \geq \theta_1$*Table Entry)
0	N/A	4.40
1	N/A	5.79
2	N/A	7.18
3	0.70	8.56
4	2.08	9.94
5	3.48	11.34
6	4.86	12.72
7	6.24	14.10
8	7.63	15.49
9	9.02	16.88
10	10.40	18.26
11	11.79	19.65
12	13.18	20.60
13	14.56	20.60
14	15.94	20.60
15	17.34	20.60
16	20.60	N/A

Table 3.6.3-7: Failure-Free Execution Interval Test Plans (Reference 2) for Failure Rate

Producer's Risk (α) (1)	**Consumer's Risk (β) (2)**	**Discrim. Ratio (d) (3)**	**Lower Test Time (λ_1T) (4)**	**Upper Test Time (λ_0T) (5)**	**Ratio of Fail-Free to Total Time (t/T) (6)**	**Expected Test Time over Total Time (ETT/T) when $\lambda = \lambda_1$ (7)**	**Expected Test Time over Total Time (ETT/T) when $\lambda = \lambda_0$ (8)**
0.10	0.10	2.442	63.308	25.925	0.10	0.88	0.43
0.10	0.10	2.814	38.581	13.710	0.15	0.84	0.45
0.20	0.20	1.793	54.330	30.301	0.10	0.84	0.52
0.20	0.20	1.968	32.618	16.574	0.15	0.81	0.53
0.20	0.20	2.147	22.445	10.454	0.20	0.78	0.54
0.20	0.20	2.338	16.640	7.117	0.25	0.76	0.55
0.20	0.20	2.547	12.927	5.075	0.30	0.73	0.56
0.20	0.20	2.779	10.365	3.730	0.35	0.71	0.58
0.20	0.20	3.052	8.501	2.785	0.40	0.68	0.59
0.30	0.30	1.438	48.707	33.871	0.10	0.80	0.59
0.30	0.30	1.695	14.361	8.473	0.25	0.74	0.61
0.30	**0.30**	**1.995**	**7.088**	3.553	**0.40**	0.68	0.62
0.30	0.30	2.454	4.086	1.665	0.55	0.62	0.63
0.30	0.30	3.059	2.526	0.826	0.70	0.58	0.66

1. The test time, T, is obtained by either dividing Column 4 by λ_1 or Column 5 by λ_2
2. After "T" is obtained, the duration of the failure-free interval, t, is calculated by multiplying Column 6 by T
3. The Expected Test Time (ETT) is dependent on the true failure rate, λ, which is unknown:
   a. When the true failure rate is λ_1, ETT is found by multiplying Column 7 by T
   b. When the true failure rate is λ_0, ETT is found by multiplying Column 8 by T

Example:

The customer specifies the lower acceptable failure rate (λ_1) as 0.0001 failures per hour. Both the Consumer's and Producer's Risk are set at 30%. The specified reliability goal for the software (λ_0) is 0.00005 failures per hour.

- The discrimination ratio (λ_1/λ_0) is calculated as (0.0001/0.00005) = 2.0
- Entering the table at α = 0.30, β = 0.30 and d = 1.995 provides λ_1T = 7.088
- Dividing λ_1T by λ_1 results in T = (7.088/0.0001) = 70,880 hours
- Since t/T = 0.40, the resulting duration of the failure-free interval, t, is (70880*0.40) = 28,352 hours

Table 3.6.3-8: Fixed-Length RDT/RQT Plans (Reference 3) for MTBF

Nominal Decision Risks		**Discrimination Ratio**	**Test Duration**	**Test Duration**	**Accept-Reject Failure Criteria**	
α	**β**	**(θ_0/θ_1)**	**(Multiples of θ_1)**	**(Multiples of θ_0)**	**Reject (Equal or More)**	**Accept (Equal or Less)**
0.10	0.10	1.5	45.0	30.0	37	36
0.10	0.20	1.5	29.9	19.9	26	25
0.10	**0.20**	**1.5**	**21.5**	14.3	18	**17**
0.10	0.10	2.0	18.8	9.4	14	13
0.10	0.20	2.0	12.4	6.2	10	9
0.20	0.20	2.0	7.8	3.9	6	5
0.10	0.10	3.0	9.3	3.1	6	5
0.10	0.20	3.0	5.4	1.8	4	3
0.20	0.20	3.0	4.3	1.4	3	2
0.30	0.30	1.5	8.0	5.3	7	6
0.30	0.30	2.0	3.7	1.9	3	2
0.30	0.30	3.0	1.1	0.37	1	0

Example:

The customer specifies the lower acceptable MTBF (θ_1) as 500 hours. The Consumer's Risk is set at 20% and the Producer's Risk is set at 10%. The specified reliability goal for the software (θ_0) is 750 hours.

- The discrimination ratio (θ_0/θ_1) is calculated as (750/500) = 1.5
- Entering the table at α = 0.10, β = 0.20 and d = 1.5 provides a test length multiplier of 21.5 based on the lower test MTBF (θ_1)
- The duration of the fixed-length test is calculated as (21.5*500) = 10,750 hours
- In order for the test to pass, there must be 17 or fewer failures

Table 3.6.3-9: PRST RDT/RQT Plans (Reference 3) for MTBF

Nominal Decision Risks		**Discrimination Ratio**	**Time to Accept Decision In MTBF (Multiples of θ_1)**			**Time to Accept Decision In MTBF (Multiples of θ_0)**		
α	**β**	**(θ_0/θ_1)**	**Min.**	**Expected**	**Max.**	**Min.**	**Expected**	**Max.**
0.10	0.10	1.5	6.60	26.0	49.5	4.40	17.3	33.0
0.20	0.20	1.5	4.19	11.4	21.9	2.79	7.60	14.6
0.10	**0.10**	**2.0**	**4.40**	**10.2**	**20.6**	2.20	5.10	10.3
0.20	0.20	2.0	2.80	4.80	9.74	1.40	2.40	4.87
0.10	0.10	3.0	3.75	6.00	10.4	1.25	2.00	3.45
0.20	0.20	3.0	2.67	3.42	4.50	0.89	1.14	1.50
0.30	0.30	1.5	3.15	5.10	6.80	2.10	3.40	4.53
0.30	0.30	2.0	1.72	2.60	4.50	0.86	1.30	2.25

Example:

The customer specifies a lower acceptable MTBF (θ_1) as 600 hours. The Consumer's Risk and the Producer's Risk are both set at 10%. The specified reliability goal for the software (θ_0) is 1200 hours.

- The discrimination ratio (θ_0/θ_1) is calculated as (1200/600) = 2.0
- Entering the table at α = 0.10, β = 0.10 and d = 2.0 indicates that, based on the lower test MTBF (θ_1), the minimum time to an accept decision is (4.40*600) = 2,640 hours
- Based on θ_1, the expected time to an accept decision is (10.2*600) = 6,120 hours
- Based on θ_1, the maximum time to an accept decision is (20.6*600) = 12,360 hours

Table 3.6.3-10: Factors for Calculating Confidence Intervals Around Test MTBF

(Assumption of Exponential Distribution)

d	Lower Limit						Upper Limit					
	99% Two-Sided 99.5% One-Sided	**98% Two-Sided 99% One-Sided**	**95% Two-Sided 97.5% One-Sided**	**90% Two-Sided 95% One-Sided**	**80% Two-Sided 90% One-Sided**	**60% Two-Sided 80% One-Sided**	**60% Two-Sided 80% One-Sided**	**80% Two-Sided 90% One-Sided**	**90% Two-Sided 95% One-Sided**	**95% Two-Sided 97.5% One-Sided**	**98% Two-Sided 99% One-Sided**	**99% Two-Sided 99.5% One-Sided**
2	0.185	0.217	0.272	0.333	0.433	0.619	4.47	9.46	19.4	39.6	100	200
4	0.135	0.151	0.180	0.210	0.257	0.334	1.21	1.88	2.83	4.10	6.67	10.0
6	0.108	0.119	0.139	0.159	0.188	0.234	0.652	0.909	1.22	1.61	2.31	3.01
8	0.0909	0.100	0.114	0.129	0.150	0.181	0.437	0.573	0.733	0.921	1.21	1.48
10	0.0800	0.0857	0.0976	0.109	0.125	0.149	0.324	0.411	0.508	0.600	0.789	0.909
12	0.0702	0.0759	0.0856	0.0952	0.107	0.126	0.256	0.317	0.383	0.454	0.555	0.645
14	0.0635	0.0690	0.0765	0.0843	0.0948	0.109	0.211	0.257	0.305	0.355	0.431	0.500
16	0.0588	0.0625	0.0693	0.0760	**0.0848**	0.0976	0.179	0.215	0.251	0.290	0.345	0.385
18	0.0536	0.0571	0.0633	0.0693	0.0769	0.0878	0.156	0.184	0.213	0.243	0.286	0.322
20	0.0500	0.0531	**0.0585**	**0.0635**	0.0703	0.0799	0.137	0.158	0.184	**0.208**	0.242	0.270
22	0.0465	0.0495	0.0543	0.0589	0.0648	0.0732	0.123	0.142	0.162	0.182	0.208	0.232
24	0.0439	0.0463	0.0507	0.0548	0.0601	0.0676	0.111	0.128	0.144	0.161	0.185	0.200
26	0.0417	0.0438	0.0476	0.0513	0.0561	0.0629	0.101	0.116	0.130	0.144	0.164	0.178
28	0.0392	0.0413	0.0449	0.0483	0.0527	0.0588	0.0927	0.106	0.118	0.131	0.147	0.161
30	0.0373	0.0393	0.0425	0.0456	0.0496	0.0551	0.0856	0.0971	0.108	0.119	0.133	0.145
32	0.0355	0.0374	0.0404	0.0433	0.0469	0.0519	0.0795	0.0899	0.0997	0.109	0.122	0.131
34	0.0339	0.0357	0.0385	0.0411	0.0445	0.0491	0.0742	0.0834	0.0925	0.101	0.113	0.122
36	0.0325	0.0342	0.0367	0.0392	0.0423	0.0466	0.0696	0.0781	0.0899	0.0939	0.104	0.111
38	0.0311	0.0327	0.0351	0.0375	0.0404	0.0443	0.0656	0.0732	0.0804	0.0874	0.0971	0.103
40	0.0299	0.0314	0.0337	0.0359	0.0386	0.0423	0.0619	0.0689	0.0756	0.0820	0.0901	0.0968

Notes:
1. d = degrees of freedom
2. For failure-truncated tests, d = 2*(number of failures accumulated when the test was terminated
3. For time-truncated tests (i.e., the number of failures is less than the total number of items initially placed on test), d = 2*(number of failures accumulated at test termination + 1)
4. Multiply the value shown in the table by the total hours on test to get MTBF figures in hours. Total hours on test = (number of items on test)*(number of test hours for each item)

<u>Example 1</u>: **Failure-Truncated Test, with Replacement**

Twenty items are placed on test until 10 failures are observed. The tenth failure occurs at 80 hours. What is the mean life of the items, and the one-sided and two-sided 95% confidence intervals for the MTBF?

- Mean life = ((20 items)*(80 hours per item))/10 failures = 160 hours
- From the table, for d = 2*10 = 20, the two-sided, lower 95% confidence factor = 0.0585
  for d = 2*10 = 20, the two-sided, upper 95% confidence factor = 0.208
  for d = 2*10 = 20, the one-sided, lower 95% confidence factor = 0.0635

Multiplying these factors by (20*80 =) 1600 total test hours results in a 95% confidence interval that the true MTBF is between 94 and 333 hours, and a 95% lower confidence limit that the true MTBF is at least 102 hours.

<u>Example 2</u>: Time-Truncated Test, without Replacement

Twenty items are placed on test for 100 hours, with 7 failures occurring at the 10, 16, 17, 25, 31, 46 and 65-hour points. What is the one-sided lower 90% confidence limit?

- Total item test hours = 10 + 16 + 17 + 25 + 31 + 46 + 65 + (13 non-failed items*100 hour per item) = 1510 hours
- The MTBF = 1510 hours/7 failures = 216 hours
- From the table, for d = 2*(7+1) = 16, the one-sided, lower 90% confidence factor = 0.0848

Multiplying this factor by 1510 test hours results in a 90% lower confidence limit that the true MTBF is greater than 128 hours.

For More Information:

1. Coppola, A., "Practical Statistical Tools for the Reliability Engineer", Reliability Information Analysis Center, September 1999
2. Lakey, P.B. and Neufelder, A.M., "System and Software Reliability Assurance Notebook", Rome Laboratory, RL-TR-97-XX, 1997
3. MIL-HDBK-781, "Handbook for Reliability Test Methods, Plans, and Environments for Engineering, Development, Qualification and Production", April 1996
4. Musa, J.D., "Software Reliability Engineering: More Reliable Software, Faster Development and Testing", McGraw-Hill, July 1998, ISBN 0079132715

Table 7.5.2-11: Summation of Terms of Poisson's Exponential Binomial Limit

Exp Fail	r																					
λt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0.02	0.980	0.020																				
0.04	0.961	0.038	0.001																			
0.06	0.942	0.056	0.002																			
0.08	0.923	0.074	0.003																			
0.10	0.905	0.090	0.005																			
0.15	0.861	0.129	0.009	0.001																		
0.20	0.819	0.163	0.017	0.001																		
0.25	0.779	0.195	0.024	0.002																		
0.30	0.741	0.222	0.033	0.004																		
0.35	0.705	0.246	0.043	0.006																		
0.40	0.670	0.268	0.054	0.007	0.001																	
0.45	0.638	0.287	0.064	0.010	0.001																	
0.50	0.607	0.303	0.076	0.012	0.002																	
0.55	0.577	0.317	0.088	0.016	0.002																	
0.60	0.549	0.329	0.099	0.020	0.003																	
0.65	0.522	0.339	0.111	0.024	0.003	0.001																
0.70	0.497	0.347	0.122	0.028	0.005	0.001																
0.75	0.472	0.355	0.132	0.034	0.006	0.001																
0.80	0.449	0.360	0.144	0.038	0.008	0.001																
0.85	0.427	0.364	0.154	0.044	0.009	0.002																
0.90	0.407	0.365	0.165	0.050	0.011	0.002																
0.95	0.387	0.367	0.175	0.055	0.013	0.003																
1.00	0.368	0.368	0.184	0.061	0.015	0.003	0.001															
1.10	0.333	0.366	0.201	0.074	0.021	0.004	0.001															
1.20	0.301	0.362	0.216	0.087	0.026	0.006	0.002															
1.30	0.273	0.354	0.230	0.100	0.032	0.009	0.002															
1.40	0.247	0.345	0.241	0.113	0.040	0.011	0.002	0.001														
1.50	0.223	0.335	0.251	0.125	0.047	0.015	0.003	0.001														
1.60	0.202	0.322	0.258	0.138	0.055	0.018	0.005	0.001														
1.70	0.183	0.310	0.264	0.150	0.063	0.022	0.006	0.002														

Table 7.5.2-11: Summation of Terms of Poisson's Exponential Binomial Limit (continued)

Exp Fail	r																					
λt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1.8	0.165	0.298	0.268	0.160	0.073	0.026	0.007	0.002	0.001													
1.9	0.150	0.284	0.270	0.171	0.081	0.031	0.010	0.002	0.001													
2.0	0.135	0.271	0.271	0.180	0.090	0.036	0.013	0.004	0.001													
2.2	0.111	0.244	0.268	0.196	0.109	0.047	0.018	0.005	0.002													
2.4	0.091	0.217	0.262	0.209	0.125	0.060	0.024	0.009	0.002	0.001												
2.6	0.074	0.193	0.251	0.218	0.141	0.074	0.032	0.012	0.004	0.001												
2.8	0.061	0.170	0.238	0.223	0.156	0.087	0.041	0.016	0.006	0.001	0.001											
3.0	0.050	0.149	0.224	0.224	0.168	0.101	0.050	0.022	0.008	0.003	0.001											
3.2	0.041	0.130	0.209	0.223	0.178	0.114	0.060	0.028	0.011	0.004	0.002											
3.4	0.033	0.114	0.103	0.218	0.186	0.127	0.071	0.035	0.015	0.005	0.002	0.001										
3.6	0.027	0.099	0.177	0.212	0.191	0.138	0.083	0.042	0.019	0.008	0.003	0.001										
3.8	0.022	0.085	0.162	0.204	0.195	0.148	0.093	0.051	0.024	0.010	0.004	0.001	0.001									
4.0	0.018	0.073	0.147	0.195	0.195	0.156	0.104	0.060	0.030	0.013	0.005	0.002	0.001									
4.2	0.015	0.063	0.132	0.185	0.194	0.163	0.114	0.069	0.036	0.017	0.007	0.003	0.001									
4.4	0.012	0.054	0.119	0.174	0.192	0.169	0.124	0.078	0.043	0.021	0.009	0.004	0.001									
4.6	0.010	0.046	0.106	0.163	0.188	0.173	0.132	0.087	0.050	0.026	0.012	0.005	0.002	0.001								
4.8	0.008	0.040	0.095	0.152	0.182	0.175	0.140	0.096	0.058	0.031	0.015	0.006	0.003	0.001								
5.0	0.007	0.034	0.084	0.140	0.175	0.175	0.146	0.104	0.065	0.036	0.018	0.008	0.003	0.001								
5.2	0.005	0.029	0.074	0.130	0.168	0.175	0.151	0.112	0.073	0.042	0.022	0.010	0.004	0.002	0.001							
5.4	0.004	0.024	0.066	0.119	0.160	0.173	0.155	0.120	0.081	0.048	0.026	0.013	0.006	0.002	0.001							
5.6	0.004	0.021	0.058	0.108	0.141	0.170	0.158	0.127	0.089	0.055	0.030	0.016	0.007	0.003	0.001							
5.8	0.003	0.017	0.051	0.098	0.143	0.165	0.160	0.133	0.096	0.062	0.036	0.019	0.009	0.004	0.001	0.001						
6.0	0.002	0.015	0.045	0.089	0.134	0.161	0.161	0.138	0.103	0.069	0.041	0.022	0.011	0.005	0.002	0.001						
6.2	0.002	0.012	0.039	0.081	0.125	0.155	0.160	0.142	0.110	0.076	0.047	0.026	0.014	0.006	0.003	0.001						
6.4	0.002	0.011	0.034	0.072	0.116	0.149	0.158	0.145	0.116	0.082	0.052	0.031	0.016	0.008	0.004	0.002	0.001					
6.6	0.001	0.009	0.030	0.065	0.107	0.142	0.156	0.147	0.121	0.089	0.059	0.035	0.019	0.010	0.005	0.002	0.001					
6.8	0.001	0.007	0.026	0.058	0.099	0.135	0.153	0.149	0.126	0.095	0.064	0.040	0.023	0.012	0.006	0.003	0.001					
7.0	0.001	0.006	0.022	0.052	0.091	0.128	0.149	0.149	0.130	0.101	0.070	0.045	0.026	0.014	0.007	0.003	0.001	0.001				
7.2	0.001	0.005	0.019	0.046	0.083	0.120	0.144	0.148	0.134	0.107	0.077	0.050	0.030	0.017	0.009	0.004	0.002	0.001				

Table 7.5.2-11: Summation of Terms of Poisson's Exponential Binomial Limit (continued)

Exp Fail	r																					
λt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
7.4	0.001	0.005	0.017	0.041	0.076	0.113	0.139	0.147	0.136	0.112	0.083	0.056	0.034	0.020	0.010	0.005	0.002	0.001				
7.6	0.001	0.003	0.014	0.037	0.070	0.106	0.134	0.145	0.138	0.117	0.089	0.061	0.039	0.023	0.012	0.006	0.003	0.001	0.001			
7.8	0.000	0.003	0.012	0.032	0.063	0.099	0.128	0.143	0.139	0.121	0.094	0.067	0.043	0.026	0.014	0.007	0.004	0.002	0.001			
8.0	0.000	0.002	0.010	0.029	0.057	0.092	0.122	0.139	0.139	0.124	0.099	0.072	0.048	0.029	0.017	0.009	0.004	0.002	0.001			
8.2	0.000	0.002	0.009	0.025	0.041	0.084	0.115	0.136	0.139	0.127	0.104	0.077	0.052	0.033	0.019	0.010	0.005	0.003	0.001			
8.4	0.000	0.002	0.008	0.022	0.047	0.078	0.109	0.131	0.138	0.129	0.108	0.082	0.058	0.037	0.022	0.013	0.006	0.003	0.001	0.001		
8.6	0.000	0.002	0.007	0.019	0.042	0.072	0.103	0.127	0.137	0.130	0.112	0.088	0.063	0.042	0.025	0.015	0.008	0.004	0.002	0.001		
8.8	0.000	0.001	0.006	0.017	0.038	0.066	0.097	0.122	0.134	0.131	0.116	0.092	0.068	0.046	0.029	0.017	0.009	0.005	0.002	0.001		
9.0	0.000	0.001	0.005	0.015	0.034	0.061	0.091	0.117	0.132	0.132	0.118	0.098	0.072	0.050	0.032	0.019	0.011	0.006	0.003	0.001	0.001	
9.2	0.000	0.001	0.004	0.013	0.030	0.055	0.085	0.111	0.128	0.131	0.120	0.101	0.077	0.055	0.036	0.022	0.013	0.007	0.003	0.002	0.001	
9.4	0.000	0.001	0.004	0.011	0.027	0.050	0.080	0.106	0.125	0.131	0.123	0.105	0.082	0.059	0.040	0.025	0.014	0.008	0.004	0.002	0.001	
9.6	0.000	0.001	0.003	0.009	0.023	0.045	0.073	0.101	0.121	0.129	0.124	0.108	0.087	0.064	0.044	0.028	0.017	0.009	0.005	0.002	0.001	0.001
9.8	0.000	0.000	0.002	0.009	0.021	0.041	0.068	0.095	0.117	0.127	0.125	0.111	0.091	0.068	0.048	0.031	0.019	0.011	0.006	0.003	0.001	0.001
10.0	0.000	0.000	0.002	0.007	0.018	0.037	0.063	0.090	0.112	0.125	0.125	0.114	0.094	0.073	0.052	0.035	0.022	0.013	0.007	0.004	0.002	0.001

Table 7.5.2-12: Summation of Terms of Poisson's Exponential Binomial Limit

Exp Fail	r																						
λt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0.02	0.980	1.000																					
0.04	0.961	0.999	1.000																				
0.06	0.942	0.998	1.000																				
0.08	0.923	0.997	1.000																				
0.10	0.905	0.995	1.000																				
0.15	0.861	0.990	0.999	1.000																			
0.20	0.819	0.982	0.999	1.000																			
0.25	0.779	0.974	0.998	1.000																			
0.30	0.741	0.963	0.996	1.000																			
0.35	0.705	0.951	0.994	1.000																			
0.40	0.670	0.938	0.992	0.999	1.000																		
0.45	0.638	0.925	0.989	0.999	1.000																		
0.50	0.607	0.910	0.986	0.998	1.000																		
0.55	0.577	0.894	0.982	0.998	1.000																		
0.60	0.549	0.878	0.977	0.997	1.000																		
0.65	0.522	0.861	0.972	0.996	0.999	1.000																	
0.70	0.497	0.844	0.966	0.994	0.999	1.000																	
0.75	0.472	0.827	0.959	0.993	0.999	1.000																	
0.80	0.449	0.809	0.953	0.991	0.999	1.000																	
0.85	0.427	0.791	0.945	0.989	0.998	1.000																	
0.90	0.407	0.772	0.937	0.987	0.998	1.000																	
0.95	0.387	0.754	0.929	0.984	0.997	1.000																	
1.00	0.368	0.736	0.920	0.981	0.996	0.999	1.000																
1.1	0.333	0.699	0.900	0.974	0.995	0.999	1.000																
1.2	0.301	0.663	0.879	0.966	0.992	0.998	1.000																
1.3	0.273	0.627	0.857	0.957	0.989	0.998	1.000																
1.4	0.247	0.592	0.833	0.946	0.986	0.997	0.999	1.000															
1.5	0.223	0.558	0.809	0.934	0.981	0.996	0.999	1.000															
1.6	0.202	0.525	0.783	0.921	0.976	0.994	0.999	1.000															
1.7	0.183	0.493	0.757	0.907	0.970	0.992	0.998	1.000															
1.8	0.165	0.463	0.731	0.891	0.964	0.990	0.999	1.000															
1.9	0.150	0.434	0.704	0.875	0.956	0.987	0.997	0.999	1.000														
2.0	0.135	0.406	0.677	0.857	0.947	0.983	0.996	0.999	1.000														
2.2	0.111	0.355	0.623	0.819	0.928	0.975	0.993	0.998	1.000														
2.4	0.091	0.308	0.570	0.779	0.904	0.964	0.988	0.997	0.999	1.000													
2.6	0.074	0.267	0.518	0.736	0.877	0.951	0.983	0.995	0.999	1.000													

Table 7.5.2-12: Summation of Terms of Poisson's Exponential Binomial Limit (continued)

Exp Fail λt	r																						
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2.8	0.061	0.231	0.469	0.692	0.848	0.935	0.976	0.992	0.998	0.999	1.000												
3.0	0.050	0.199	0.423	0.647	0.815	0.916	0.966	0.988	0.996	0.999	1.000												
3.2	0.041	0.171	0.380	0.603	0.781	0.895	0.955	0.983	0.994	0.998	1.000												
3.4	0.033	0.147	0.340	0.558	0.744	0.871	0.942	0.977	0.992	0.997	0.999	1.000											
3.6	0.027	0.126	0.303	0.515	0.706	0.844	0.927	0.969	0.988	0.996	0.999	1.000											
3.8	0.022	0.107	0.269	0.473	0.668	0.816	0.909	0.960	0.984	0.994	0.998	0.999	1.000										
4.0	0.018	0.092	0.238	0.433	0.629	0.785	0.889	0.949	0.979	0.992	0.997	0.999	1.000										
4.2	0.015	0.078	0.210	0.395	0.590	0.753	0.867	0.936	0.972	0.989	0.996	0.999	1.000										
4.4	0.012	0.066	0.185	0.359	0.551	0.720	0.844	0.921	0.964	0.985	0.994	0.998	0.999	1.000									
4.6	0.010	0.056	0.163	0.326	0.513	0.686	0.818	0.905	0.955	0.980	0.992	0.997	0.999	1.000									
4.8	0.008	0.048	0.143	0.294	0.476	0.651	0.791	0.887	0.944	0.975	0.990	0.996	0.999	1.000									
5.0	0.007	0.040	0.125	0.265	0.440	0.616	0.762	0.867	0.932	0.968	0.986	0.995	0.998	0.999	1.0000								
5.2	0.006	0.034	0.109	0.238	0.406	0.581	0.732	0.845	0.918	0.960	0.982	0.993	0.997	0.999	1.000								
5.4	0.005	0.029	0.095	0.213	0.373	0.546	0.702	0.822	0.903	0.951	0.977	0.990	0.996	0.999	1.000								
5.6	0.004	0.024	0.082	0.191	0.342	0.512	0.670	0.797	0.886	0.941	0.972	0.988	0.995	0.998	0.999	1.000							
5.8	0.003	0.021	0.072	0.170	0.313	0.478	0.638	0.771	0.867	0.929	0.965	0.984	0.993	0.997	0.999	1.000							
6.0	0.002	0.017	0.062	0.151	0.285	0.446	0.606	0.744	0.847	0.916	0.957	0.980	0.991	0.996	0.998	0.999	1.000						
6.2	0.002	0.015	0.054	0.134	0.259	0.414	0.574	0.716	0.826	0.902	0.949	0.975	0.989	0.995	0.998	0.999	1.000						
6.4	0.002	0.012	0.046	0.119	0.235	0.384	0.542	0.687	0.803	0.886	0.939	0.969	0.986	0.994	0.997	0.999	1.000						
6.6	0.001	0.010	0.040	0.105	0.213	0.355	0.511	0.658	0.780	0.869	0.927	0.963	0.982	0.992	0.997	0.999	0.999	1.000					
6.8	0.001	0.009	0.034	0.093	0.192	0.327	0.480	0.628	0.755	0.850	0.915	0.955	0.978	0.990	0.996	0.998	0.999	1.000					
7.0	0.001	0.007	0.030	0.082	0.173	0.301	0.450	0.599	0.729	0.830	0.901	0.947	0.973	0.987	0.994	0.998	0.999	1.000					
7.2	0.001	0.006	0.025	0.072	0.156	0.276	0.420	0.569	0.703	0.810	0.887	0.937	0.967	0.984	0.993	0.997	0.999	0.999	1.000				
7.4	0.001	0.005	0.022	0.063	0.140	0.253	0.392	0.539	0.676	0.788	0.871	0.926	0.961	0.980	0.991	0.996	0.998	0.999	1.000				
7.6	0.001	0.004	0.019	0.055	0.125	0.231	0.365	0.510	0.648	0.765	0.854	0.915	0.954	0.976	0.989	0.995	0.998	0.999	1.000				
7.8	0.000	0.004	0.016	0.048	0.112	0.210	0.338	0.481	0.620	0.741	0.835	0.902	0.945	0.971	0.986	0.993	0.997	0.999	1.000				
8.0	0.000	0.003	0.014	0.042	0.100	0.191	0.313	0.453	0.593	0.717	0.816	0.888	0.936	0.966	0.983	0.992	0.996	0.998	0.999	1.000			
8.5	0.000	0.002	0.009	0.030	0.074	0.150	0.256	0.386	0.523	0.653	0.763	0.849	0.909	0.949	0.973	0.986	0.993	0.997	0.999	0.999	1.000		
9.0	0.000	0.001	0.006	0.021	0.055	0.116	0.207	0.324	0.456	0.587	0.706	0.803	0.876	0.926	0.959	0.978	0.989	0.995	0.998	0.999	1.000		
9.5	0.000	0.001	0.004	0.015	0.040	0.089	0.165	0.269	0.392	0.522	0.645	0.752	0.836	0.898	0.940	0.967	0.982	0.991	0.996	0.998	0.999	1.000	
10.0	0.000	0.000	0.003	0.003	0.029	0.067	0.130	0.220	0.333	0.458	0.583	0.697	0.792	0.864	0.917	0.951	0.973	0.986	0.993	0.997	0.998	0.999	1.000

Section 4.0: Test Support Activities

INSIGHT

A successful software reliability test program requires two fundamental activities: data collection and analyses and failure analysis. A rigorous, closed-loop failure analysis process must be in place to ensure that all potential defects discovered during software testing are properly analyzed for relevance and impact to the targeted users, root cause determined, and corrective action developed, implemented and verified. Without a strong failure analysis system, it is highly likely that defects will be overlooked and/or corrective actions will be less than effective or create other problems. The lack of a rigorous failure analysis process will result in wasted resources, both time and money. Defects which would have been detected early in the development process will be passed along to the customer, where they will become more costly to correct. Likewise, a sound data collection and analysis and failure analysis process down to root cause will ensure that the proper conclusions will be drawn from the testing process.

Topic 4.1: Failure Reporting and Corrective Action System (FRACAS)

Topic 4.1.1: FRACAS Overview

A **F**ailure **R**eporting, **A**nalysis and **C**orrective **A**ction **S**ystem (FRACAS) is one of the most critical elements in the development, implementation, operation and maintenance process through which the reliability of software or a system can be continually improved. An effective FRACAS should always capture:

(1) Failure reporting information through which an historical trend or reliability growth database can be created
(2) The steps taken during a failure analysis, and the results obtained, to be able to determine the root cause of the failure
(3) The documented corrective action that, once implemented and verified, eliminates or mitigates the reoccurrence of the failure

The concept of a formalized FRACAS has traditionally been applied to hardware products/systems, but it can be effectively applied to all types of products (including software and service) and processes (i.e., manufacturing, billing, design, administrative, etc.). The basic measure of FRACAS effectiveness is its ability to function as a closed-loop coordinated system in the identification and correction of failure modes related to product/process, and the identification, implementation and verification of a corrective action to preclude recurrence of the failure. As a result, early elimination of failure modes/mechanisms or trends is a major contributor to reliability growth and continuous process improvement.

The key points to consider in implementing a FRACAS, and defining how formal or complex it should be, are:

- FRACAS has been publicly acknowledged as a major success element for many types of products, and for many different kinds of companies
- It is absolutely essential to reach mutual agreement with your customer(s) or end-user(s) on the definition of "fault" and "failure" before development and testing begin, preferably built right into the specification
- FRACAS can be used effectively to capture, analyze and correct failure modes at any point in the system life cycle, from development to retirement
- There is no cookbook approach or cost-benefit optimization model that defines what is an effective FRACAS for all industries or for all applications
- Tailoring of the FRACAS should be considered mandatory if it is to be successful
- If the FRACAS is not closed-loop (providing feedback for action and approval by all appropriate personnel), then it will not be effective
- The data collected by the FRACAS should be no less than that required to cost-effectively identify and correct root failure causes, but no more than what will be realistically available and useful, given resource constraints (people, time and money)
- The better the case that can be made for proving that FRACAS will provide long-term life cycle cost benefits for the company, the more likely that upper management will support its use
- The ultimate purpose of FRACAS should be to meet customer needs and expectations through improved system performance and reliability
- Improved user satisfaction, system performance and continued reliability growth will lead to lower operating costs, improved competitive position, and larger market share

The overall effectiveness of the FRACAS will always be defined by the accuracy and completeness of the data captured in the initial report that documents a failure or fault. The initial problem or trouble report should describe, as a minimum:

- ***Who*** discovered the fault/failure (by name or operator number)

- ***What*** specifically failed, and what was the observed indication of the fault/failure (what were the symptoms)
- ***Where*** did it fail (in the lab; during test; at the end-user's site; during a critical mission?)
- ***When*** did it fail (date; time of day; shift)
- Under what conditions did it fail (operational; applied environmental stresses; sequence of preceding events; etc.)

The ***Why*** of the failure, and the ***How*** of avoiding its occurrence in the future, can only be successfully determined through a detailed analysis of the information available from the initial report.

Figure 4.1.1-1 illustrates a feedback loop for the occurrence of failures at various stages of a system life cycle. At each stage of development, the closed-loop FRACAS should capture and assess information regarding each failure incident, as illustrated in Figure 4.1.1-2 and outlined in Table 4.1.1-1. Figure 4.1.1-3 illustrates an example Failure Analysis Report form. Tables 4.1.1-2 and 4.1.1-3 provide an overview of common failure modes and failure classifications, respectively.

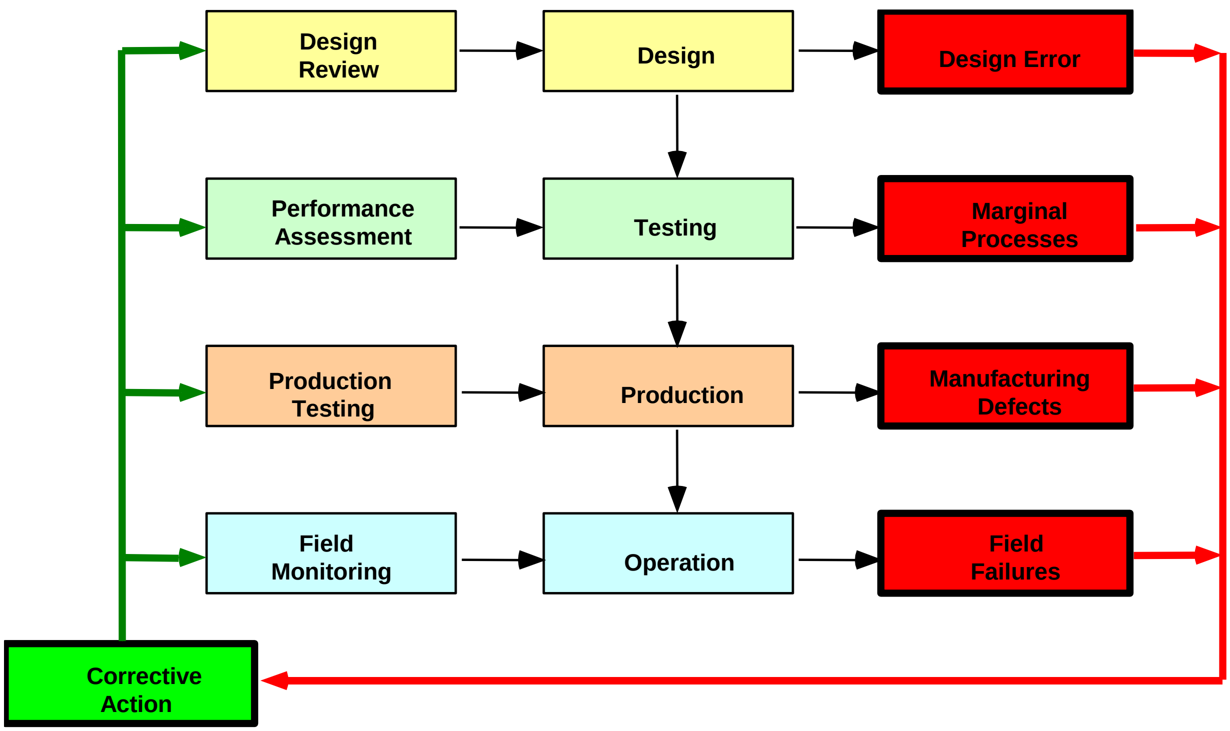
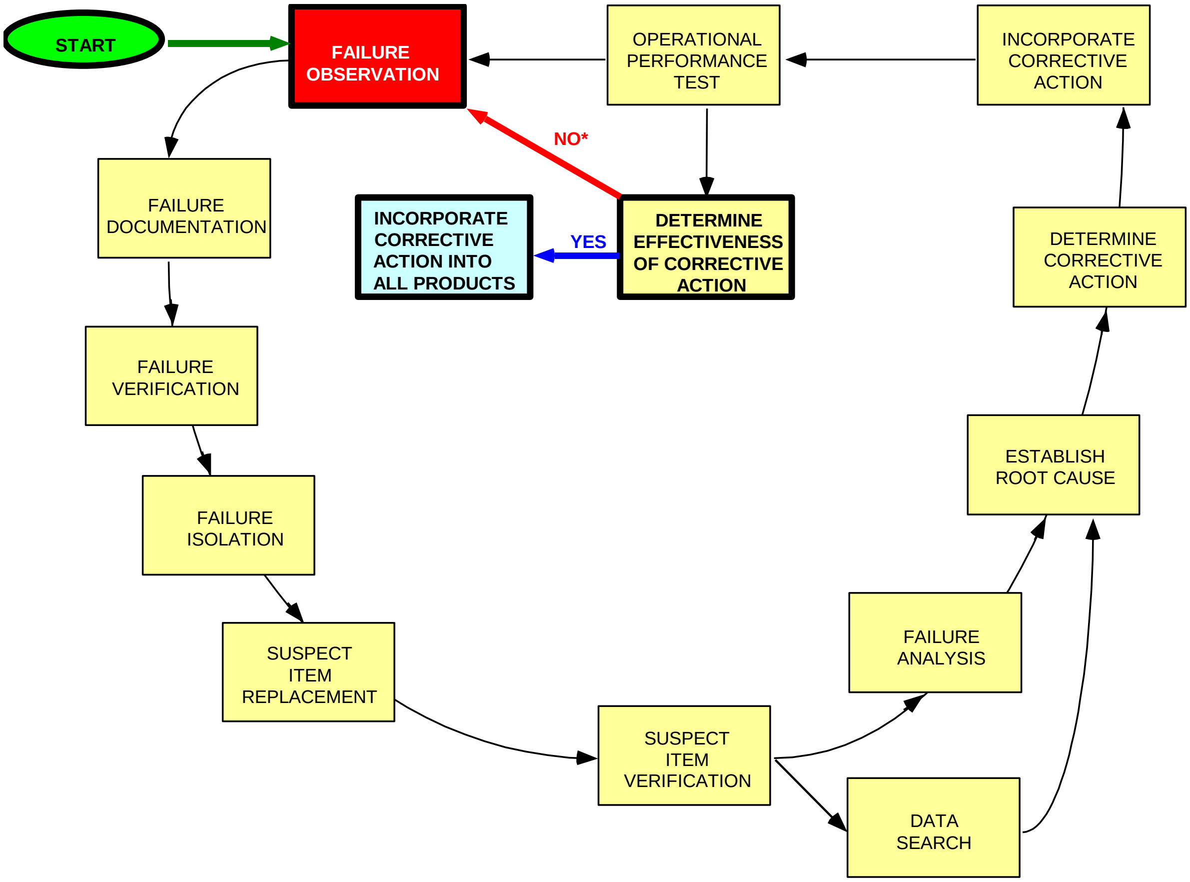


Figure 4.1.1-1: Representative Feedback Loop for a Product Life Cycle



* NOTE: If the corrective action is not effective then the proper root cause may not have been identified, and the failure will continue to occur.

Figure 4.1.1-2: Closed-Loop FRACAS

Table 4.1.1-1: Steps for a Successful Failure Analysis

Step	Action	Rationale
Fault/Failure Observation	Identify that a fault/failure has occurred and notify proper personnel	Operating conditions that resulted in the fault/failure should be maintained until they have been reviewed, if possible
Fault/Failure Documentation	Record all data related to the conditions leading up to the fault/failure	Pertinent data includes a concise description of the fault/failure, supporting data, and the sequence of events
Fault/Failure Verification	Verify fault/failure by repeating events causing fault/failure	Repetition helps discern between “hard” failures and those caused by operator or procedural errors
Fault/Failure Isolation	Perform additional testing and troubleshooting to isolate the cause of the fault/failure	A fault/failure may be isolated to a defective design, infant mortality, wear-out, or external causes (operator error, support equipment failures, or improper procedures)
Suspect Item Replacement	For verified faults/failures, replace the suspect part, assembly or software with a known good item or corrected code. Recreate the conditions causing the fault/failure, and the tests detecting them, to confirm suspect item replacement. If fault/failure repeats, repeat fault isolation activity to determine correct cause.	The end item, once proven to be functional following suspect item replacement, returns to its development/ manufacturing process. Any replaced hardware should be "tagged" for repair. The configuration of faulty software should be documented. "Tagging" should include all information relative to the incident. It should also allow for documentation of subsequent failure analysis and corrective action activities.

Table 4.1.1-1: Steps for a Successful Failure Analysis (continued)

Step	Action	Rationale
Suspect Item Verification	Verify failure of the item independent of the system. If it cannot be verified, review previous verification/isolation activities to ensure that the proper cause of the fault/failure has been identified.	Isolation to lower levels of system structure is critical to find root cause. Inability to verify a failure may result from an inability to recreate the sequence of events or identify interaction dependencies.
Data Search	In parallel with failure analysis, search the FRACAS database and other databases for failure trends/patterns for identical or similar items	Hardware failures may result from a defective lot of parts or poor process quality. Software failures may relate to defective code from a supplier possibly due to version upgrades, or from OSS. Searches outside the FRACAS database (e.g., bulletin boards, technical literature) may identify problems experienced by others.
Failure Analysis	Determine from data search results and criticality of the failure how extensive the analysis should be. Perform the analysis to a level low enough to determine its root cause.	Determining factors should be (1) short-term costs vs. long-term savings, (2) schedule impact vs. customer satisfaction, (3) warranty costs vs. liability costs. Analysis should also identify external contributing factors.
Establish Root Cause	Determine the initial, basic condition that was the direct cause of the failure (i.e., if the condition hadn't existed, the failure would never have occurred)	Root-cause analysis places greater emphasis on failure mode elimination or prevention, relying on an understanding of the architecture and interfaces of the defective item that precipitated the failure.
Determine Corrective Action	Based on the root failure cause, develop, document and communicate a corrective action (CA) that may prevent the failure from reoccurring	Corrective action should emphasize long-term solutions that address the root cause, not band-aid fixes. Action can include redesign or selection of different suppliers.
Incorporate Corrective Action	Incorporate the identified CA in the failed product as a minimum, pending verification of its effectiveness	Delays in incorporating CA means additional defective items may be delivered. Large-scale incorporation, however, should not occur until the CA has been verified. Timing should be based on confidence that the root failure cause has been eliminated or satisfactorily mitigated.
Operational Performance Test	After CA is incorporated, perform baseline tests and operational tests to verify proper functionality under static and dynamic conditions. Compare all results to pre-failure data to identify potential shifts in baseline data.	Testing under normal or "accelerated" conditions should be performed to provide confidence that the failure has been eliminated, or its effects minimized. Future faults/failures not related to the implemented CA should be considered new FRACAS events.
Determine Corrective Action Effectiveness	Verify that the CA has (1) corrected the original fault/failure, and (2) not introduced additional fault/failures or degraded system operation below acceptable threshold levels. If the original fault/failure reoccurs, the FRACAS process must be repeated.	A CA is not effective if it introduces other faults/failures or degrades performance to unacceptable levels. A CA is not effective if testing has not instilled confidence that the fault/failure has been eliminated or satisfactorily mitigated. Effectiveness should be tracked through future system performance.
Incorporate Corrective Action Globally	Expand the proven CA into the product population (subject to retrofit considerations). Track, document and report future fault/failures indicating lack of CA effectiveness.	Design-related CAs should be tracked to ensure CAs for different future faults/failures do not degrade the effectiveness of the original CA, or that the original CA did not introduce new failure modes that will result in future faults/failures.

<table>
<tr><td colspan="3"></td><td colspan="4">FAILURE ANALYSIS REPORT</td><td colspan="2">1. NO.</td><td colspan="2">2. PAGE 1 of _____</td></tr>
<tr><td colspan="2">3. PROJECT NAME OR NUMBER</td><td>4. SYSTEM</td><td>5. SERIAL NO.</td><td>6. ENVIRONMENT/TEST LEVEL</td><td>7. MALFUNCTION DATE</td><td colspan="2">8. OPERATING HOURS/CYCLES</td><td colspan="3">9. REPORTED BY</td></tr>
<tr><td>MAJOR COMPONENT OR UNIT</td><td colspan="2">10. NAME</td><td>11. REF. DES.</td><td>12. PART NO.</td><td colspan="4">13. MANUFACTURER</td><td colspan="2">14. SERIAL NO.</td></tr>
<tr><td>SUBASSEMBLY</td><td colspan="2">15. NAME</td><td>16. REF. DES.</td><td>17. PART NO.</td><td colspan="4">18. MANUFACTURER</td><td colspan="2">19. SERIAL NO.</td></tr>
<tr><td>SUBASSEMBLY</td><td colspan="2">20. NAME</td><td>21. REF. DES.</td><td>22. PART NO.</td><td colspan="4">23. MANUFACTURER</td><td colspan="2">24. SERIAL NO.</td></tr>
<tr><td>PART</td><td colspan="2">25. NAME</td><td>26. REF. DES.</td><td>27. PART NO.</td><td>28. GENERIC NO.</td><td colspan="4">29. MANUFACTURER</td><td>30. SERIAL NO./ DATE CODE</td></tr>
<tr><td colspan="11">31. RELATED FAILURE REPORT NUMBERS</td></tr>
<tr><td colspan="11">32. HISTORY</td></tr>
<tr><td colspan="11">33. ANALYSIS</td></tr>
<tr><td colspan="11">34. CONCLUSIONS</td></tr>
<tr><td colspan="11">35. CORRECTIVE ACTION/RECOMMENDATIONS</td></tr>
<tr><td colspan="4">36. CORRECTIVE ACTION VERIFICATION BY</td><td colspan="3">37. DOCUMENT NO.</td><td colspan="4">38. EFFECTIVITY</td></tr>
<tr><td colspan="2">39. PREPARED BY</td><td>DATE</td><td colspan="3">40. APPROVAL (RELIABILITY)</td><td>DATE</td><td colspan="4">41. PROBLEM NO.</td></tr>
<tr><td colspan="2">42. APPROVAL (ENGINEERING)</td><td>DATE</td><td colspan="3">43. APPROVAL (PROGRAM)</td><td>DATE</td><td colspan="4">44. DISTRIBUTION</td></tr>
</table>

Figure 4.1.1-3: Example Failure Analysis Report Form

Table 4.1.1-2: Failure Categories

Failure Category	Description
Equipment Manufacturer Design	Any failure which can be traced directly to the design of the product
Equipment Manufacturer Workmanship	Any failure which is caused by poor workmanship or inadequate process controls during product construction, inspection, testing or repair
Part Manufacturer Design	Any failure which can be traced directly to the design of the part causing it to fail or degrade resulting in the product failure
Part Manufacturer Workmanship	Any part failure which is caused by poor workmanship or inadequate process controls during part construction, inspection, testing or repair and which subsequently results in product failure
Software Error	**A product failure caused by an error in the software programming associated with the function of the product**
Test Operator Error	A product failure associated with a mistake in performing steps of a test procedure. The product itself does not fail, or fails due to induced conditions imposed by the operator error (secondary failure).
Test Procedure Error	A product failure associated with an improperly written test procedure. The product itself does not fail, or fails due to induced conditions imposed by the test procedure error (secondary failure).
Test Equipment Error	A failure associated with the failure of supporting test equipment, which can include environmental support equipment, or support equipment used to supply electrical/mechanical stimuli or measure product operational performance
Secondary Failure	A product failure which damages/degrades product parts, resulting from (1) a relevant part failure within the product which induces additional part failures or (2) induced product part failures resulting from test operator, test procedure, or test equipment errors

Table 4.1.17.7-3: Failure Classifications

Failure Classifications	Description
Failure, Relevant	A product (or service) failure which has been verified and can be expected to occur in normal operational use
Failure, Non-Relevant	A product (or service) failure which has been verified as having been caused by a condition not defined for normal operational use
Failure, Chargeable	A relevant primary failure of the product (or service) under test, and any secondary failures resulting from a single failure incident
Failure, Non-Chargeable	A non-relevant failure, or a relevant failure caused by a previously agreed to set of conditions which eliminates the assignment of failure responsibility to a specific functional group
Failure, Pattern	The occurrence of two or more failures of the same part (or function) in identical or equivalent applications, where the failures are caused by the same basic failure mechanism, and the failures occur at a rate inconsistent with the expected part (or function) failure rate
Failure, Multiple	Simultaneous occurrence of two or more verified independent failures. When two or more failed parts are found during troubleshooting, and assignable causes cannot be verified as dependent, multiple failures are presumed to have occurred.

Tailoring the FRACAS, and the extent to which root-cause analysis and corrective action should be pursued given dollar, resource and schedule constraints, should be based on classification of faults/failures into logical groups that can help set priorities to effectively identify corrective actions. Table 4.1.1-4 provides an outline for tailoring root-cause analysis and corrective action priorities based on the defined criticality of an expected fault impact on the system.

Table 4.1.1-4: Setting Root-Cause Analysis and Corrective Action Priorities

Priority Level	**Criteria for Pursuing Corrective Action**
1	Applies if a fault could (1) prevent the accomplishment of a capability, or (2) jeopardize safety, security, or any other requirement identified as "critical"
2	Applies if a fault could (1) adversely affect the accomplishment of a capability, or (2) adversely affect technical, cost or schedule risks to the project, or to life-cycle support of the system. In either case, **no** workaround solution is known.
3	Applies if a fault could (1) adversely affect the accomplishment of a capability, or (2) adversely affect technical, cost or schedule risks to the project, or life-cycle support of the system. In either case, a workaround solution is known
4	Applies if a fault could (1) result in user/operator inconvenience or annoyance, but does not affect a required capability, or (2) result in inconvenience or annoyance for development or support personnel, but does not prevent the accomplishment of their responsibilities
5	Applies if a fault results in any other effect not covered under priorities 1 through 4

Classification of failures into pre-defined categories can help in the summarization of data to review failure history and identify failure trends. A simple classification scheme for software is given in Table 4.1.1-5. A more detailed classification scheme from IEEE 1044-1993 (Reference 1) that includes categories for Disposition and Impact, would be considered appropriate for large-scale development efforts for safety-critical systems, or for pursuit of CMMI Level 5 certification. A slightly modified summary of this classification that recognizes potential human factors is provided in Table 4.1.1-6.

Table 4.1.1-5: General Categories for Classifying Software Problems

Category	**Applies to problems in…**
Plans	Any of the plans developed for the project
Concept	The operational concept
Requirements	The system or software requirements
Design	The design of the system or software
Code	The software code
Database/Data File	A database or data file
Test Information	Test plans, test descriptions, or test products
Manuals	The user, operator or support manuals
Other	Other software products

Table 4.1.1-6: Summary of IEEE 1044-1993 Software Anomaly Classifications

Category	Classifications	Subclassifications
RECOGNITION		
Project Activity	Analysis; Review; Audit; Inspection; Code/Compile/Assemble; Testing; Validation/Qualification Testing; Support/Operational; Walk-Through	
Project Phase	Requirements	Concept Evaluation; System Requirements; Software Requirements; Prototype Requirements
	Design	System Design; Preliminary Design; Detail Design; Prototype Design
	Implementation	Code; Unit Test; Integrate; Prototype
	Test	Integration Test; System Test; Beta Test; Prototype Test; Acceptance Test; Installation and Checkout
	Operation and Maintenance	
	Retirement	
Suspected Cause	Product	Hardware; Software; Human Factors; Data; Interface; Documentation; Enhancement (Perceived Inadequacies)
	Test System	Hardware; Software; Human Factors; Data; Interface; Documentation; Enhancement (Perceived Inadequacies)
	Platform	Hardware; Operating System; Human Factors; Documentation
	Outside Vendor/Third Party	Hardware; Software; Human Factors; Data; Documentation; Enhancement (Perceived Inadequacies)
	User	
	Unknown	
Repeatability	One Time Occurrence; Intermittent; Recurring; Reproducible; Unknown	
Symptom	Operating System Crash	
	Program Hang-Up	
	Program Crash	
	Input Problem	Correct Input Not Accepted; Wrong Input Accepted; Description Incorrect or Missing; Parameters Incomplete or Missing; Wrong Format; Incorrect Result/Data; Incomplete/Missing; Spelling/Grammar; Cosmetic
	Output Problem	Wrong Format; Incorrect Result/Data; Incomplete/Missing; Spelling/Grammar; Cosmetic
	Failed Required Performance	
	Perceived Total Product Failure	
	System Error Message	
	Other	
Product Status	Usable; Degraded; Affected, Use Workaround; Unaffected	
INVESTIGATION		
Actual Cause	Product	Hardware; Software; Human Factors; Data; Interface; Documentation; Enhancement (Perceived Inadequacies)
	Test System	Hardware; Software; Human Factors; Data; Interface; Documentation; Enhancement (Perceived Inadequacies)
	Platform	Hardware; Operating System; Human Factors; Documentation
	Outside Vendor/Third Party	Hardware; Software; Human Factors; Data; Documentation; Enhancement (Perceived Inadequacies)
	User	
	Unknown	
Source	Specification	Requirements; Functional; Preliminary Design; Detailed Design; Product Design; Interface; Data; Implementation
	Code	
	Database	
	Manuals and Guides	User Guide; Reference Manual; Product Internal Training Manual; System Administrator Manual; Installation Guide
	Plans and Procedures	Test Plan; Test Procedures; Quality Assurance Plan; Configuration Management Plan; Maintenance Plan; Product Support Plan
	Reports	Test Report; Quality Assessment Report
	Standards/Policies	

Table 4.1.1-6: Summary of IEEE 1044-1993 Software Anomaly Classifications (continued)

Category	Classifications	Subclassifications
INVESTIGATION (continued)		
Type	Logic Problem	Forgotten Case or Steps; Duplicate Logic; Extreme Conditions Neglected; Unnecessary Function; Misinterpretation; Missing Condition Test; Checking Wrong Variable; Iterating Loop Incorrectly
	Computation Problem	Equation Insufficient or Incorrect; Precision Loss; Sign Conversion Fault
	Interface/Timing Problem	Interrupts Handled Incorrectly; I/O Timing Incorrect; Subroutine/Module Mismatch
	Data Handling Problem	Initialized Data Incorrectly; Accessed or Stored Data Incorrectly; Scaling or Units of Data Incorrect; Dimensioned Data Incorrectly; Scope of Data Incorrect
	Data Problem	Sensor Data Incorrect or Missing; Operator Data Incorrect or Missing; Embedded Data in Tables Incorrect or Missing; External Data Incorrect or Missing; Output Data Incorrect or Missing; Input Data Incorrect or Missing
	Documentation Problem	Ambiguous Statement; Incomplete Item; Incorrect Item; Missing Item; Conflicting Items; Redundant Items; Confusing Items; Illogical Item; Unverifiable Item; Unachievable Item
	Document Quality Problem	Application Standards Not Met; Not Traceable; Not Current; Incomplete; Inconsistencies
	Enhancement	Change in Program Requirements; Improve Comments; Improve Code Efficiency; Implement Editorial Changes; Improve Usability; Software Fix of a Hardware Problem; Other Enhancement
	Failure Caused by Previous Fix	
	Performance Problem	
	Interoperability Problem	
	Standards Conformance Problem	
	Other Problem	
ACTION		
Resolution	Immediate	Software Fix; Update Project Documentation; Operator Training; Test Software Fix; Outside Vendor/Third Party
	Eventual	Software Fix; Update Project Documentation; Operator Training; Test Software Fix; Outside Vendor/Third Party
	Deferred	Fix in Later Release; Waiver Requested (Reference)
	No Fix	No Problem Found; Waiver Requested (Reference); Fix Not Justifiable; Fix Not Identifiable; Obsolete
Corrective Action	Department Action	Revise Process (Policies/Procedures); Implement Training; Create/Revise/Reinforce Standards/Specifications; Reallocate People/Resources; Improve/Enforce Audit Activities
	Corporate Action	Revise Process (Policies/Procedures); Implement Training; Create/Revise/Reinforce Standards/Specifications; Reallocate People/Resources; Improve/Enforce Audit Activities
	Industry/Government	Sponsor Research/Education Programs; Compile/Publish Data; Create/Revise/Reinforce Standards/Specifications; Improve/Enforce Audit Activities
	Institutions for Research/Education	Research Problem; Develop New Technologies; Test Alternate Approaches; Create/Revise Tests; Enforce Educational Standards
DISPOSITION		
Disposition	Closed	Resolution Implemented; Not a Problem; Not in Project Scope (Unresolvable); Outside Vendor's Problem (Reference); Duplicate Problem (Reference)
	Deferred (Reference)	
	Merged with Another Problem (Reference)	
	Referred to Another Project (Reference)	
IMPACT		
Severity	Urgent; High; Medium; Low; None	
Priority	Urgent; High; Medium; Low; None	
Customer Value	Priceless; Critical; High; Medium; Low; None; Detrimental	
Mission Safety	Urgent; High; Medium; Low; None	
Project Schedule	High; Medium; Low; None	
Project Cost	High; Medium; Low; None	

Table 4.1.1-6: Summary of IEEE 1044-1993 Software Anomaly Classifications (continued)

Category	Classifications	Subclassifications
IMPACT (continued)		
Project Risk	High; Medium; Low; None	
Project Quality/Reliability	High; Medium; Low; None	
Societal	High; Medium; Low; None	

For More Information:

1. Tsung, P.W., "An Extended Implementation of FRACAS," Society of Automotive Engineers, Communications in RMS, Vol. 1, No. 1, 1994.
2. Magnus, J.S., "Standardized FRACAS for Non-Standardized Products," 1989 Proceedings Annual R&M Symposium, 1989.
3. "A Reliability Guide to Failure Reporting, Analysis and Corrective Action System," American Society for Quality Control, 1977.
4. IEEE 1044-1993 "Standard Classification for Software Anomalies"
5. IEEE 1044.1-1995 "Guide to Classification for Software Anomalies"
6. Neufelder, A.M., "Ensuring Software Reliability", Marcel Dekker, Inc., 1993, ISBN 0824787625
7. Nicholls, D.; "Failure Reporting, Analysis and Corrective Action System (FRACAS) Application Guidelines", Reliability Information Analysis Center, FRACAS, September 1999

Topic 4.1.2: Orthogonal Defect Classification

Orthogonal Defect Classification (ODC) is a methodology and framework which can be used as part of a defect prevention program to classify and tag software defects into predefined defect classes throughout the development and operational lifecycle. ODC then provides techniques for performing measurement and analysis of the data gathered to gain insight and provide feedback to developers and managers on the progress of a project. Managers can then take proactive measures based on what the ODC data is saying.

ODC essentially involves categorizing a defect into a particular class that collectively points to the part of the process which needs attention. ODC extracts semantics of defects based on a classification scheme. The classification scheme provides information about progress against a project lifecycle. Examining the change in distribution of defects over lifecycle phases allows the manager to measure progress against the lifecycle.

Other defect classification techniques, such as identifying where the defect was inserted, may be error-prone since it forces the programmer to guess where the error was inserted. Furthermore, if the process changes, then the data is invalid. The ODC semantic classification is invariant to process and product.

ODC techniques involves, for each defect, identifying by the developer or tester each defect's **type** and **trigger**.

Defect Types

Defect types are assigned to each defect by the software developer who makes the repair to the software to fix the defect. Furthermore the software developer defines whether a defect was caused by something *missing* or something *incorrect*. Defect types, as shown in Table 4.1.2-1, are intended to be simple and obvious to the software developer, with little room for incorrect assignment or confusion.

Table 4.1.2-1: Defect Type Classification Scheme

Defect Type	Defect Description	Life Cycle Phase(s) Where Defect Type is Associated. Verification/Testing Activities Where Defect Should be Found
Function	Defect that affects capability, end-user features, product interfaces, hardware architecture, or global data structures. This type of defect requires a formal design change	Design. Found at Design Review
Assignment	Defect caused by incorrect data structure or control block initialization. Typically involves changing or repairing a few lines of code. These type of defects should be found in code reviews or unit tests	Coding Phase. Detected in Code Reviews and Unit Tests
Interface	Defect caused by errors in interacting with other components, modules, device drivers, etc.	Detected in Systems Integration Tests.
Checking	Defect caused by improper data or variable validation before used, in conditional statements, or in loop conditions in logic	Coding Phase. Detected in Code Reviews and Unit Tests
Timing and Serialization	Defect caused by improper management of shared and real-time resources	
Build, package, merge	Defects in library systems, management of changes, or version control	
Documentation	Defects in publications and other maintenance information	
Algorithm or Logic	Defects in an algorithms efficiency or correctness which can be fixed by (re)implementing an algorithm or local data structure without a design change.	Low Level Design. Detected in Design Reviews

Defect Triggers

Whereas defect types are able to measure development progress within the system lifecycle, defect triggers are used to measure verification/testing progress in the software development lifecycle. Defect triggers are what caused the fault to surface and result in a failure. There are three classes of software triggers associated with the types of verification or defect detection method that occur:

- Review and Inspection tests – identifies problems in a product through a human review of design documents and code. This would include inspections. This class of trigger occurs by humans thinking about factors such as design conformance. The quality of defects identified is tied to the skill level of the human. See Table 4.1.2-2 for details of review test trigger types.
- Unit/Function tests – identifies problems by execution of the software code. Test plans are designed and written to uncover such things as functional completeness. Each test case has a trigger associated with it. See Table 4.1.2-3 for details of review test trigger types.
- System tests – identifies problems by emulating usage under customer environmental conditions. System testing attempts to uncover defects that are likely to be found in the field. This type of test is typically performed when most of the software is available. This type of test stresses the products through increased workload or changing the software configurations. See Table 4.1.2-4 for details of system test trigger types.

Table 4.1.2-2: Review and Inspection Triggers

Defect Trigger Type	Trigger Description
Backward compatibility	Defect related to how the current version of the software previous versions of the software or in anticipation of future releases
Lateral compatibility	Defect related to how this subsystem would work with other subsystems within the same software configuration.
Design conformance	Defect related to the completeness of the product with respect to the requirements and overall goals of the product.
Concurrency	Defect related to the serialization and timing issues in the design and implementation of the product
Operational semantics	Defect related to the logic flow within the design or implementation of a product
Document consistency/completeness	Defect related to the overall completeness of a design and consistency between the different parts of the design or implementation.
Rare situation	Defect related conditions peculiar to a product that the casual observer would not immediately recognize, such as unusual implementations, idiosyncrasies, or domain specific information that is not common.

Table 4.1.2-3: Unit/Function Test Triggers

Defect Trigger Type	Why Was The Test Case Written?	Test Type
Test coverage	Exercise a function through the various inputs to maximize the coverage possible in the parameter space.	Black Box
Test sequencing	Tests to sequence multiple bodies of code with different sequences.	Black Box
Test interaction	Tests more complicated interactions between multiple bodies of code unusually not covered by simple sequences.	Black Box
Test variation	Test a single function with multiple inputs	Black Box
Simple path coverage	Test different paths through the code, to increase code coverage	Clear box
Combination path coverage	Tests more complete code paths, exercising branches and different sequences.	Clear box

Table 4.1.2-4: System Test Triggers.

Defect Trigger Type	Trigger Description
Recovery/exception handling	Defect occurs when the exception handling or recovery process occurs because of conditions in the workload
System start-up and restart	Defect occurs when a product is initialized or being shut down from regular operation. Typically associated with maintenance operations.
Workload volume/stress	Defect occurs when the system has been stressed and reaches a resource or capability limit.
Hardware configuration and software configuration	Defect occurs when the hardware or software environment is changed.
Normal mode	Defect occurs when nothing unusual has occurred.

Analyzing ODC Data

ODC is intended to aid users in gaining more insight into the nature and cause of defects being found and corrected during development, verification and testing processes. Most of the analysis can be performed with simple spreadsheet graphing and analysis capabilities.

A typical usage is to monitor defect types over each period or phase of a project and look for unexpected patterns or trends of various defect types. Figure 4.1.2-1 (showing only 4 defect types) would represent a typical ODC phase-based graph showing the percentage of defect types found in each phase of development. The phases shown in sequence do not imply a waterfall lifecycle, but rather represent names for typical phases within a project, whether they are within sequential, incremental, spiral, agile, etc., lifecycles.

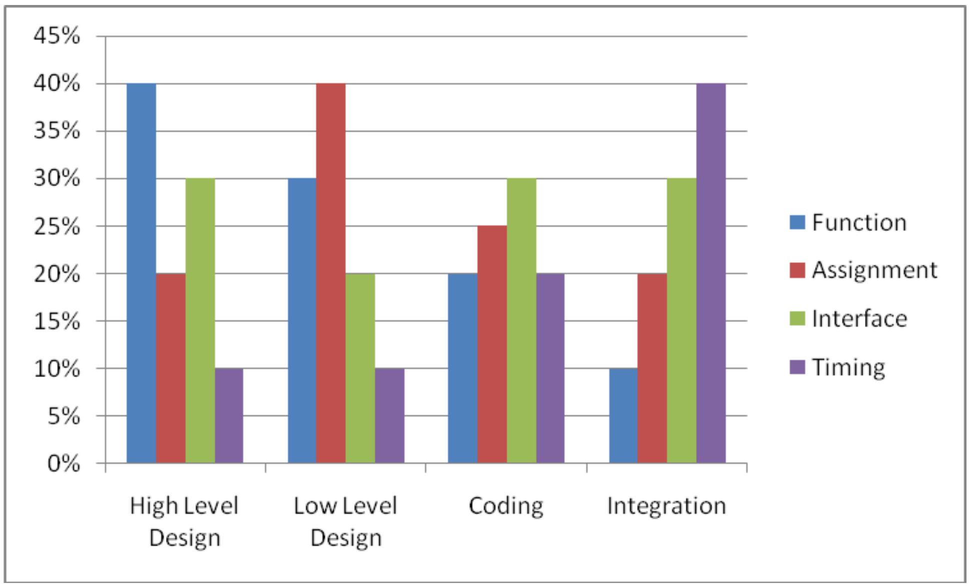


Figure 4.1.2-1 Typical Defect Type by Phase Graph

The following analysis and observations can be made about Figure 4.1.2-1:

- Function type defects are decreasing over time, which is desirable given that functional type issues should be addressed and resolved during the early design of a system. If function type defects are still high during the coding or integration phase may indicate that although the project is in the coding or integration phase, the project has not progressed past the design phase and corrective action is required.
- Timing defects are increasing and peaks during integration, which is expected given that during integration is when software operates on real hardware.
- Assignment defects should peak as part of testing during coding.
- Interface defects would be expected to peak also when on real hardware in the integration phase.

Defect trigger mechanisms can be analyzed as well, especially when combined with defect types. For example Figure 4.1.2-2 hypothetically represents the defect type results (including whether the defect represented something was wrong or missing) of a design review of a web-services interface. This design review was the review of design document(s), so the high number of documentation type defects is what would be expected. Further given this is a design review, the fact that there is a relatively large number of function and algorithm type defects are also expected.

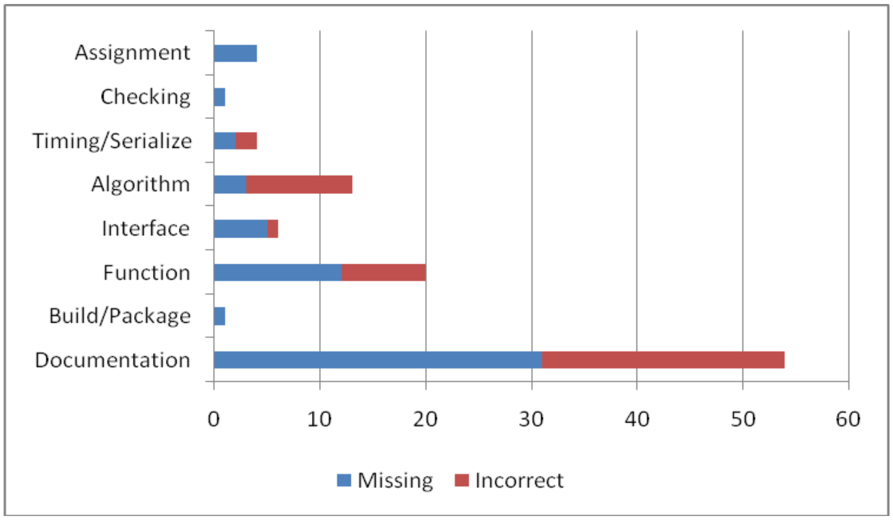


Figure 4.1.2-2 Defect Type Distribution Observed at Design Inspection

Figure 4.1.2-3, given that this defect data comes from a design inspection, shows the defect triggers that were observed. Given this is a web services interface design review, it would be expected to see in the data many lateral compatibility type triggers. However, as seen in Figure 4.1.2-3, relatively few were identified and further only function type lateral compatibility defects were found. It would have been expected that more interface type defects would have been found. One possible explanation for this result could be that the makeup of the capabilities of the inspection team was such that no one had adequate design experience; in which case others could be asked to review the design documents.

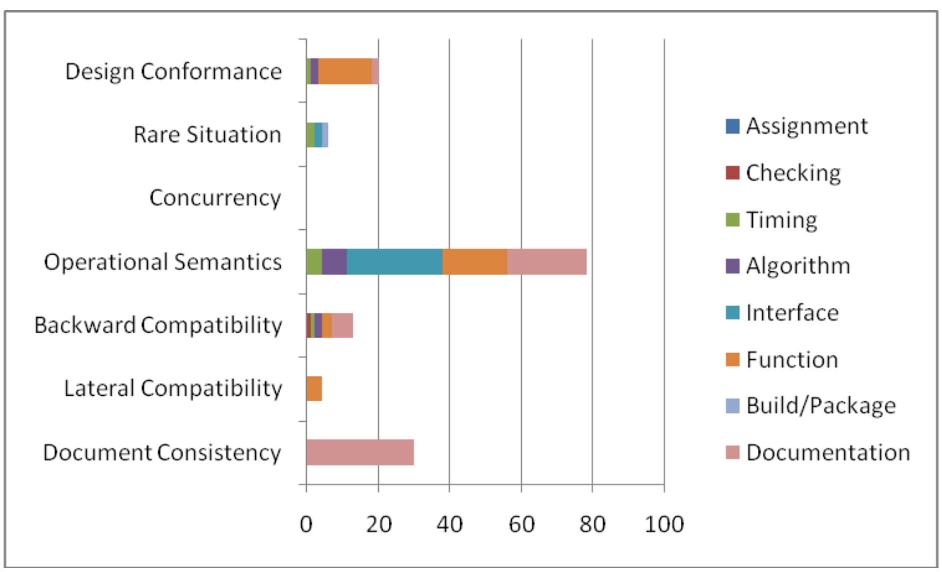


Figure 4.1.2-3 Defect Triggers Identified by the Inspection Team

ODC and Growth Models

ODC data analysis can also be used to augment typical growth models to provide more project insight. For example, consider the reliability growth model in Figure 4.1.2-4. As of Day 123, it is difficult to interpret what may be going on with the project or to predict the end of development.

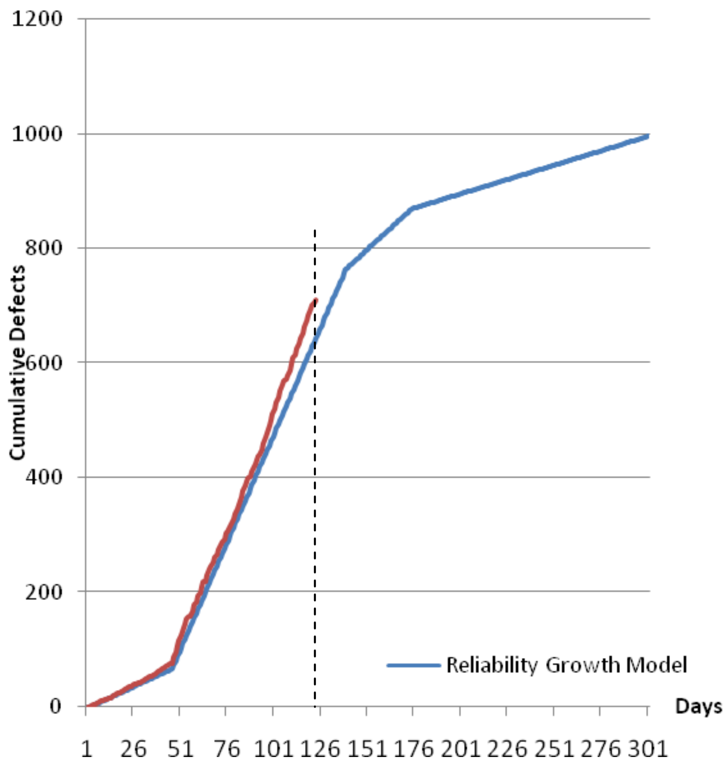


Figure 4.1.2-4 Reliability Growth Model

However, as shown in Figure 4.1.2-5, examining on the same time line various ODC defect types provides more information and hints as to what the manager should do. In this figure the function and algorithm defects have stabilized, implying that the design aspects of the pro may have stabilized. Assignment and checking defects, as a possible indicator of code quality, have not stabilized. The manager could infer then that more senior developers should be added to help stabilize the code.

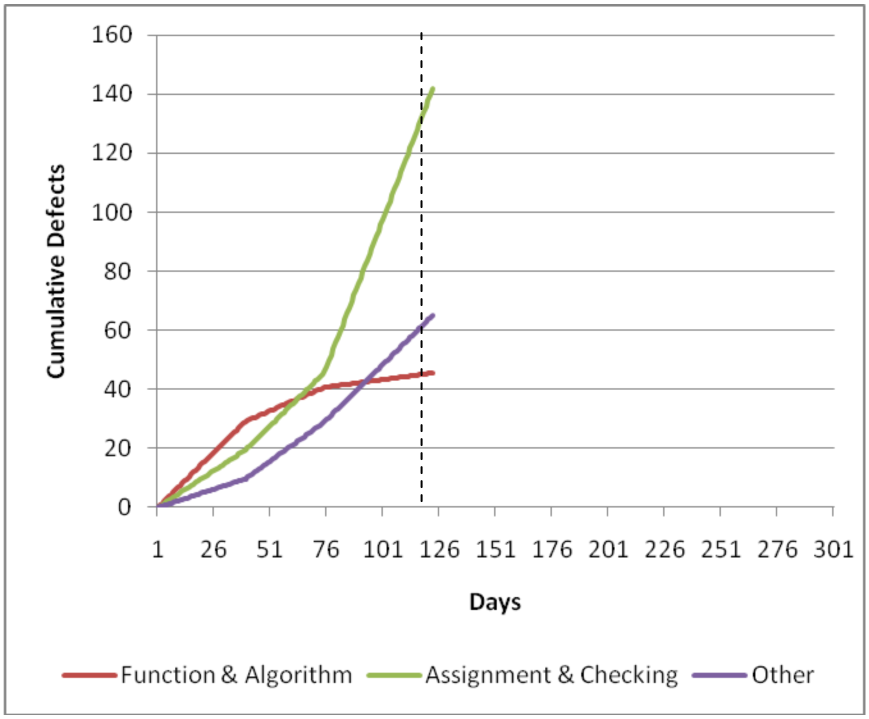


Figure 4.1.2-5. Defect Growth by Defect Type

Use in Root Cause Analysis

A defect prevention program typically involves performing root cause analysis on every defect, which can be costly on a large program. Given ODC addresses the cause and effect aspects of defects, ODC allows organizations to concentrate on root cause analysis of groups of high impact defects rather than the entire population of defects.

Implementing ODC

Implementing ODC requires:

- Modifying the defect tracking form and associated defect tracking processes to collect four additional parameters on each defect:
    - Defect Type, as described above, and whether the defect was caused by something *missing* or something *incorrect*
    - Source of the defect, such as new software, old software, reused software, etc.
    - Impact of the defect on the user
    - Defect Trigger, as described above
- Educating the developers on use and benefits of the new parameters and ODC.
- Implementing tools and educating users on analyzing the resultant data collected.
- Institutionalizing the use of ODC.

Experience from the Field

Ram Chillarege from IBM was the inventor of ODC and has experienced usage of ODC on over 50 projects (Reference 1). In Reference 2, the author claims a 10:1 cost reduction in use in root-cause analysis as well as reported a 3:1 cycle time reduction and an 80x defect reduction over a 5 year period.

Motorola (Reference 3) has reported using ODC to identify where to focus the development effort, understand the opportunities for improving the development process, understand the opportunities for improving testing, provide a system approach of causal analysis of field defects, be part of the quality management strategy.

Hewlett Packard (Reference 4) has analyzed the results of using ODC as compared to Hewlett Packard's (HP) Defect Origins, Types, and Modes. Other users of ODC include Philips Electronics India (Reference 6), Lucent (Reference 7), and others.

For More Information:

1. R. Chillarege, I.S. Bhandari, J.K. Chaar, M.J. Halliday, D.S. Moebus, B.K. Ray, and M-Y Wong, "Orthogonal Defect Classification – A Concept for In-Process Measurements", *IEEE Transactions on Software Engineering*, Nov. 1992
2. R. Chillarege, "ODC – a 10x for Root Cause Analysis", *Proceedings RAM 2006 Workshop*, Berkeley CA, May 2006
3. B. Hirsh, Motorola, "Our Experience Using Orthogonal Defect Classification", *Proceedings of International Conference on Applications of Software Measurement (ASM),* San Jose, CA., March 6-10, 2000.
4. J. Huber, Hewlett Packard, "A Comparison of IBM's Orthogonal Defect Classification to Hewlett Packard's Defect Origins, Types, and Modes", *Proceedings of International Conference on Applications of Software Measurement (ASM)*, San Jose, CA., March 6-10, 2000.
5. Michael R. Lyu (ed.), Handbook of Software Reliability Engineering, IEEE and McGraw-Hill, 1996. pp. 367-399
6. A.A. Shenvi, "Defect Prevention with Orthogonal Defect Classification," Proceedings of the 2nd Annual Conference on India Software Engineering Conference, 2009, ISBN:978-1-60558-426-3
7. N.B. Sreenivasan, Lucent Technologies, "Experiences with Orthogonal Defect Classification Technique at Lucent Technologies", *Proceedings, Fast Abstracts and Industrial Practices, The 10th International Symposium on Software Reliability Engineering (ISSRE),* Boca Raton, FL, November 1-4, 1999
8. Also see http://www.research.ibm.com/softeng/comm/odc_ext.htm

Topic 4.2: Overview of Data Collection and Analysis for Reliability Growth

The primary objective for collecting and analyzing defect and failure data is to diagnose, categorize and correct them, either in the design itself, or in the processes used to develop it. Most organizations may already collect the information that is needed to support a system or software reliability effort, but it is important to emphasize that it is not necessary to collect every bit of data regarding a project as it evolves over its life cycle. The law of diminishing returns will dictate that overly complex data collection, particularly without sufficient capability to effectively analyze the data, will result in little growth in reliability.

The types of questions that data should answer over the long term include:

- What development or maintenance process is exhibiting poor reliability and why (predominant failure modes and causes)?
- How often are these failures occurring (defect/failure rates, MTTF/MTBF)?
- How expensive is it to identify and fix these failures ($$/defect)?
- Which items are more prone to failure?
- What design or process change will most effectively detect or eliminate these failures from occurring?
- How can the effectiveness of the design or process change be quantified and verified (decreased defect/failure rates, improved product/system reliability)?

Figure 4.2-1 illustrates the steps that should be followed in setting up an effective reliability data collection and analysis process. Table 4.2-1 provides additional insight into each of these recommended steps.

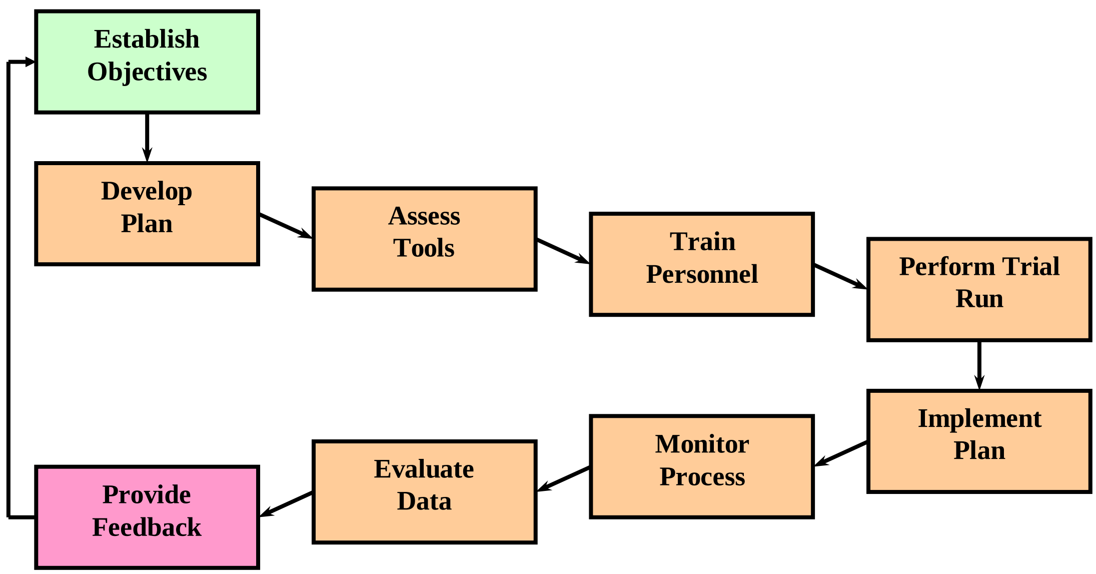


Figure 4.2-1: Overview of the Data Collection & Analysis Process

Table 4.2-1: Steps in Setting Up a Data Collection and Analysis Process

Step	Stage	Description
Establish Objectives	Planning	Accurate establishment of objectives makes the difference between successful and unsuccessful data collection efforts. Objectives would likely include product measures (e.g., size/target values of quality attributes), process measures (e.g., schedule lengths) and resource measures (e.g., development/maintenance efforts)
Develop Plan		Plan development should include all involved parties to ensure that everyone understands how the data collection/analysis tasks will be performed, and how all participating organizations will be impacted. The following questions should be addressed as part of the plan: • How often will data be gathered? • Who will gather the data? • In what form will the data be gathered? • How will the data be processed and stored? • How will the process be monitored to ensure data integrity and satisfaction of objectives? • Can existing processes capture the data and meet the objectives? • How much effort (schedule, resources) will be required to collect the necessary data over the system life cycle?
Assess Tools		The availability, maturity and usability of all data collection tools must be assessed, as well as their reliability, ease of use, robustness and support. Tools developed internally should include plans for adequate cost/schedule resources to support the development and acceptance testing of the tools.
Train Personnel		Anyone who will be using the data collection/analysis tools should be trained in their use, and must understand both the purpose of the measurements and how the supporting data will be collected. The capabilities and constraints of each tool must be understood. In addition, a common cause of invalid data is different interpretations of definitions by different people. Training helps to standardize definitions for all members of the data collection and analysis team.
Perform Trial Run		A trial run of the data plan should be carried out to precipitate and correct any problems that might result from implementation of the plan. The trial run should be carried out as early as possible in the design development phase as a means to save time and effort.
Implement Plan		At the conclusion of the planning stage, sufficient resources should have been allocated to cover the necessary staffing and tool needs, and that the required resources are available for immediate implementation.
Monitor Process	Monitoring	In order to be successful, the data collection process should be monitored on a regular basis to ensure that the objectives of the data collection and analysis process, as well as the reliability goals of the software, are being met.
Evaluate Data	Assessment	The data should be analyzed on a regular basis, starting early enough in the design and development process so that defects are detected and corrected well before delivery of the item to the customer, and preferably before entering test. Depending on the development effort, weekly evaluations may be appropriate (Reference 3). The initial collection of defect information should be validated with later information to ensure that data is accurate. The need for accuracy should be stressed to any who report and analyze the data. Once the data is validated using a comprehensive cross-section, sample data can be used to ensure that the data remains accurate. The steps involved in one type of elementary analysis of defect data are: • Sort the collected data by its defect origin (i.e., class of defects) • Count the number of defects in each group and rank them according to their criticality (highest to lowest) for successful system/process performance • For a realistic number of the top ranked items (defined through a technique such as Pareto analysis), perform a root-cause analysis to determine (1) what caused the defect, (2) what corrective action can be implemented to prevent the defect from occurring in the future or to minimize its impact, (3) how can the corrective action taken be verified as effective, i.e., it fixes the defect and doesn't introduce new defects.
Provide Feedback	Feedback	Feedback should be provided early and often during data collection and analysis throughout the systems or software life cycle, but it is especially important for closure at the end of the development effort. Everyone involved in the data collection and analysis effort should be aware of their impact on the project, particularly as it relates to the level of achieved reliability and the meeting of program objectives.

Reliability/failure data can be obtained from a number of sources, including an in-house failure reporting system; reliability test and (in the case of software) debug data; subcontractor or supplier data (if COTS/GOTS/OSS items are used); field data; and reliability data banks (which may include history on similar systems/products, or reliability experience data for reused items). Data obtained from subcontractors and suppliers may not be reliable, as some bias in the data may be present. Similarly, field data may not be as good as in-house data, since field data tends to be incomplete. Regardless of the data source, all factors that may influence the quality of the data need to be clearly understood in order for the conclusions that are drawn from the data to be credible and supportable. These factors include the ground rules for collecting the data and the assumptions made during the analysis.

From a reliability assessment viewpoint, failure data is used to:

- Determine the underlying probability distribution of time to failure and estimate its parameters (if not already known)
- Determine a point estimate of a specific reliability parameter such as mean time to failure (MTTF) or mean time between failure (MTBF)
- Determine a statistical confidence interval that is believed to contain the true value of that parameter

The two methods that are used to analyze failure data are graphical methods and statistical analysis. Graphical methods are typically the easiest to apply and produce adequate results for estimating the underlying statistical distribution in the majority of applications. Graphical methods are almost always a useful predecessor activity to more detailed statistical analysis techniques.

For field data analysis (Reference 3), the important objectives are to:

- Assess the actual quality and reliability of a product in its actual operational environment (do the field failure modes and frequency match what was expected from analytical reliability analyses and predictions/estimations)
- Determine the compliance of the field reliability data to requirements and maintenance resource planning
- Relate field failure behavior to how the item is used in the field, and to its development and maintenance processes, through the use of reliability models
- Predict product/system behavior in the field and control its field reliability by controlling the processes for its development, testing, and maintenance processes and methods

The various types of data analyses include:

- Exploratory techniques: Includes techniques in which the objective is simply to explore the potential nature of the data (plots and graphs; data modeling and associated diagnostics; data transformation; etc.)
- Confirmation techniques: Used after a body of evidence (i.e., sufficient data) has emerged to confirm or deny the popular prevailing thought (hypothesis testing; trend analysis)

The basic idea behind graphical methods is to use special probability plotting paper on which the cumulative distribution function (CDF) or the cumulative hazard function can be plotted as a straight line for the particular distribution being studied. The two parameters of the straight line (slope and intercept) allow the two parameters of the underlying distribution to be determined. The probability graph papers are based upon plots of the variable of interest (usually hours for reliability data) against the cumulative percent probability.

Data first needs to be ranked (or ordered) and the cumulative probability calculated. Order numbers are assigned based on progressive failure times. Mean ranking (when the underlying distribution is assumed to be symmetrical, as in the Normal distribution) or median ranking (when the underlying distribution is assumed to be skewed, as in the Weibull distribution) is used to determine the appropriate plotting positions of each failure on the graph paper. Table 4.2-2 illustrates a sample of 20 data points representing how data is rank-ordered, the determination of the mean and median ranking points (remember, only 1 is used), and the calculation of the CDF (i/n). Median ranks can be calculated or determined from existing tables (Table 4.2-3).

Mean Ranking:

$$\text{Mean Rank} = \mathbf{r_i} = \frac{i}{n+1}$$

where,

r_i = i^{th} order value

i = Order number

n = Total number of failure points

Median Ranking:

$$\text{Median Rank} = \mathbf{r_i} = \frac{i - 0.3}{n + 0.4}$$

where,

r_i = i^{th} order value

i = Order number

n = Total number of failure points

Table 4.2-2: Data on Times to Failure of 20 Items

Order No.	Time to Failure (hours)	Cumulative % (cdf)	Mean Rank (%) (cdf)	Median Rank (%) (cdf)
1	175	5	5	3.41
2	695	10	10	8.31
3	872	15	14	13.22
4	1250	20	19	18.12
5	1291	25	24	23.02
6	1402	30	29	27.93
7	1404	35	33	32.83
8	1713	40	38	37.74
9	1741	45	43	46.24
10	1893	50	48	47.55
11	2025	55	52	52.45
12	2115	60	57	57.36
13	2172	65	62	62.26
14	2418	70	67	67.17
15	2583	75	71	72.07
16	2725	80	76	76.98
17	2844	85	81	81.88
18	2980	90	86	86.78
19	3268	95	90	91.69
20	3538	100	95	96.59

Table 4.2-3: Table of Median Ranks (for up to 20 failures)
Sample size = n; Failure order number = i

	n																			
i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	.5000	.2929	.2063	.1591	.1294	.1091	.0943	.0830	.0741	.0670	.0611	.0561	.0519	.0483	.0452	.0424	.0400	.0378	.0358	.0341
2		.7071	.5000	.3864	.3147	.2655	.2295	.2021	.1806	.1632	.1489	.1368	.1266	.1788	.1101	.1034	.0975	.0922	.0874	.0831
3			.7937	.6136	.5000	.4218	.3648	.3213	.2871	.2594	.2366	.2175	.2013	.1873	.1751	.1644	.1550	.1465	.1390	.1322
4				.8409	.6853	.5782	.5000	.4404	.3935	.3557	.3244	.2982	.2760	.2568	.2401	.2254	.2125	.2009	.1905	.1812
5					.8706	.7345	.6352	.5596	.5000	.4519	.4122	.3789	.3506	.3263	.3051	.2865	.2700	.2553	.2421	.2302
6						.8906	.7705	.6787	.6065	.5481	.5000	.4596	.4253	.3958	.3700	.3475	.3275	.3097	.2937	.2793
7							.9057	.7979	.7129	.6443	.5878	.5404	.5000	.4653	.4350	.4085	.3850	.3641	.3453	.3283
8								.9170	.8194	.7406	.6756	.6211	.5747	.5347	.5000	.4695	.4425	.4184	.3968	.3774
9									.9259	.8368	.7634	.7018	.6494	.6042	.5650	.5305	.5000	.4728	.4484	.4264
10										.9330	.8551	.7825	.7240	.6737	.6300	.5915	.5575	.5272	.5000	.4755
11											.8389	.8632	.7987	.7432	.6949	.6525	.6150	.5816	.5516	.5245
12												.9439	.8734	.8127	.7599	.7135	.6725	.6359	.6032	.5736
13													.9481	.8822	.8249	.7746	.7300	.6903	.6547	.6226
14														.9517	.8899	.8356	.7875	.7447	.7063	.6717
15															.9548	.8966	.8450	.7991	.7579	.7207
16																.9576	.9025	.8535	.8095	.7698
17																	.9600	.9078	.8610	.8188
18																		.9622	.9126	.8678
19																			.9642	.9169
20																				.9659

Table 4.2-4 illustrates the characteristics and the steps of how to use the Normal and Weibull (of which the Exponential distribution is a special case, i.e., $\beta = 1.0$) distributions to evaluate reliability data.

Table 4.2-4: Analyzing Reliability Data

Normal Distribution		**Weibull Distribution (includes Exponential)**	
When to Use: Method estimates the mean (μ) and standard deviation (σ) of the data when failure times are normally distributed		**When to Use:** The flexibility of the Weibull distribution makes it useful for describing the probability density function for a variety of distributions (most notably for software reliability, the exponential distribution, where $\beta = 1.0$)	
Conditions for Use: Failure times must be collected, but may be censored. Normal probability paper is required		**Conditions for Use:** Failure times must be collected, but may be censored. Estimates of the Weibull shape (β) and scale (α) parameters may be obtained graphically using *ln-ln*, or special Weibull probability graph paper	
Method	**Example**	**Method**	**Example**
1. Plot the "i^{th}" failure time in a sample of "n" ordered failure times on the lower axis vs. the **mean** ranking points on the right axis	1. From Table 4.2-3, plot the failure time from Column 2 for each ordered point (x-axis) vs. its **mean** ranking point from Column 4 (y-axis).	1. Plot the "i^{th}" failure time in a sample of "n" ordered failure times on the lower axis vs. the **median** ranking points on the left axis	1. From Table 4.2-3, plot the failure time from Column 2 for each ordered point (x-axis) vs. its **median** ranking point from Column 5 (y-axis).
2. Draw the best line fit through the plotted points by using the last plotted point as the reference point and dividing the remaining points into two equal groups above and below the line	2. See Figure 4.2-2	2. Draw the best line fit through the plotted points so that an equal number of data points appear on either side of the line	2. See Figure 4.2-3
3. The mean (μ) is estimated by projecting the 50% probability of failure point to the line, then projecting that intersection down to the x-axis. The estimate of the mean ($\bar{x}$) is read off of the x-axis.	3. The value of $\bar{x}$ is read as 2000 hours	3. If the Weibull paper being used does not allow β to be read directly, then, for *ln-ln* paper calculate it as: $$\beta = \frac{\ln\ln(\frac{1}{1-\mathbf{F}(\mathbf{t}_2)}) - \ln\ln(\frac{1}{1-\mathbf{F}(\mathbf{t}_1)})}{\ln \mathbf{t}_2 - \ln \mathbf{t}_1}$$ If *log-log* paper is being used, then: $$\beta = \frac{\log\ln(\frac{1}{1-\mathbf{F}(\mathbf{t}_2)}) - \log\ln(\frac{1}{1-\mathbf{F}(\mathbf{t}_1)})}{\log \mathbf{t}_2 - \log \mathbf{t}_1}$$	3. Assuming that *ln-ln* paper has been used, and reading from the graph, $F(t_2) = 0.99$ hours, $F(t_1) = 0.02$ hours, $t_2 = 4150$ hours. $t_1 = 375$ hours. Therefore, the slope is calculated as: $$\beta = \frac{1.527-(-3.902)}{2.404}$$ $$\beta = 2.258$$
4. The standard deviation (σ) is estimated by first projecting the 84% probability of failure point to the line, then projecting that intersection down to the x-axis (Point U), then repeating this process for the 16% point (Point L). The estimate of the standard deviation is calculated as: $$\mathbf{s} = \frac{\mathbf{U}-\mathbf{L}}{2}$$	4. U = 3020 hours L = 1010 hours s = (3020-1010)/2 = 1005 hours	4. The scale parameter, α (or characteristic life), is read by first projecting the 63.2% probability of failure point to the line, then projecting that intersection down to the x-axis. The estimate of α is read off the x-axis	4. The characteristic life of the software is read from the graph as approximately 2100 hours

Table 4.2-4: Analyzing Reliability Data (continued)

Normal Distribution		Weibull Distribution (includes Exponential)	
Method	**Example**	**Method**	**Example**
5. The 95% confidence limits around the mean are given by: $$\bar{x} \pm t \frac{s}{\sqrt{n}}$$ where "t" is the student-t distribution statistic, available from lookup tables. The value of this statistic for various sample sizes, n, is shown below: n — t 5 — 2.57 10 — 2.23 20 — 2.09 30 — 2.04 50 — 2.00 : — 1.96	5. The resulting confidence limits around the mean are: $2000 \pm {(2.09)(1005)}/{\sqrt{20}}$ $2000 \pm 470 \text{hours}$	5. The reliability of the software at a specific point in time is found by drawing a vertical line up from the x-axis at a specific point in time, then horizontally projecting the line from the point of intersection to the probability of failure axis and subtracting that value from 1.00.	5. The reliability of the software at 1000 hours, as read from the graph, is (1-0.19), or 81%

A simple graphical technique that can be used to test whether collected data is represented by an exponential distribution is to plot the cumulative test or operating time against the cumulative number of failures, as illustrated in Figure 4.2-4. If the plot will support a reasonably straight line, then a constant failure rate is indicated and an exponential distribution of failures can be assumed.

Table 4.2-5 and Figure 4.2-5 illustrate the calculation of fault density, hazard rate and reliability from time interval data (length of time interval between each failure is measured). Table 4.2-6 and Figure 4.2-6 illustrate these same calculations using failure interval data (number of failures within each fixed time interval is measured). The basic formulae for each case are given below:

Table 4.2-4a: Reliability Calculations

Function	Time Interval Data	Failure Interval Data
Failure Density	$$\mathbf{f(t)} = \frac{1}{(\text{Total \# of Intervals}) \text{ x } (\text{\# of hours in Interval})}$$	$$\mathbf{f(t)} = \frac{\text{Total \# of Failures in Interval}}{(\text{Total \# of Systems}) \text{ x } (\text{\# of hours in Interval})}$$
Hazard Rate	$$\mathbf{h(t)} = \frac{1}{(n + 1 - i) \text{ x } (\text{\# of hours in Interval})}$$ where, n = total # of intervals in dataset i = interval # being evaluated	$$\mathbf{h(t)} = \frac{\text{Total \# of Failures in Interval}}{(n - \sum_{i=1}^{j} a_{i-1}) \text{ x } (\text{\# of hours in Interval})}$$ where, n = total # of "systems" in dataset i = interval # being evaluated a_i = number of failures in the i^{th} interval j = total # of intervals in dataset
Reliability	$$\mathbf{R(t)} = \frac{(\text{Total \# of Intervals}) - i}{\text{Total \# of Intervals}}$$ where, i = interval # being evaluated	$$\mathbf{R(t)} = \frac{(\text{Total \# of Systems}) - F_i}{\text{Total \# of Systems}}$$ where, F_i = cumulative # of failed "systems" through interval "i"

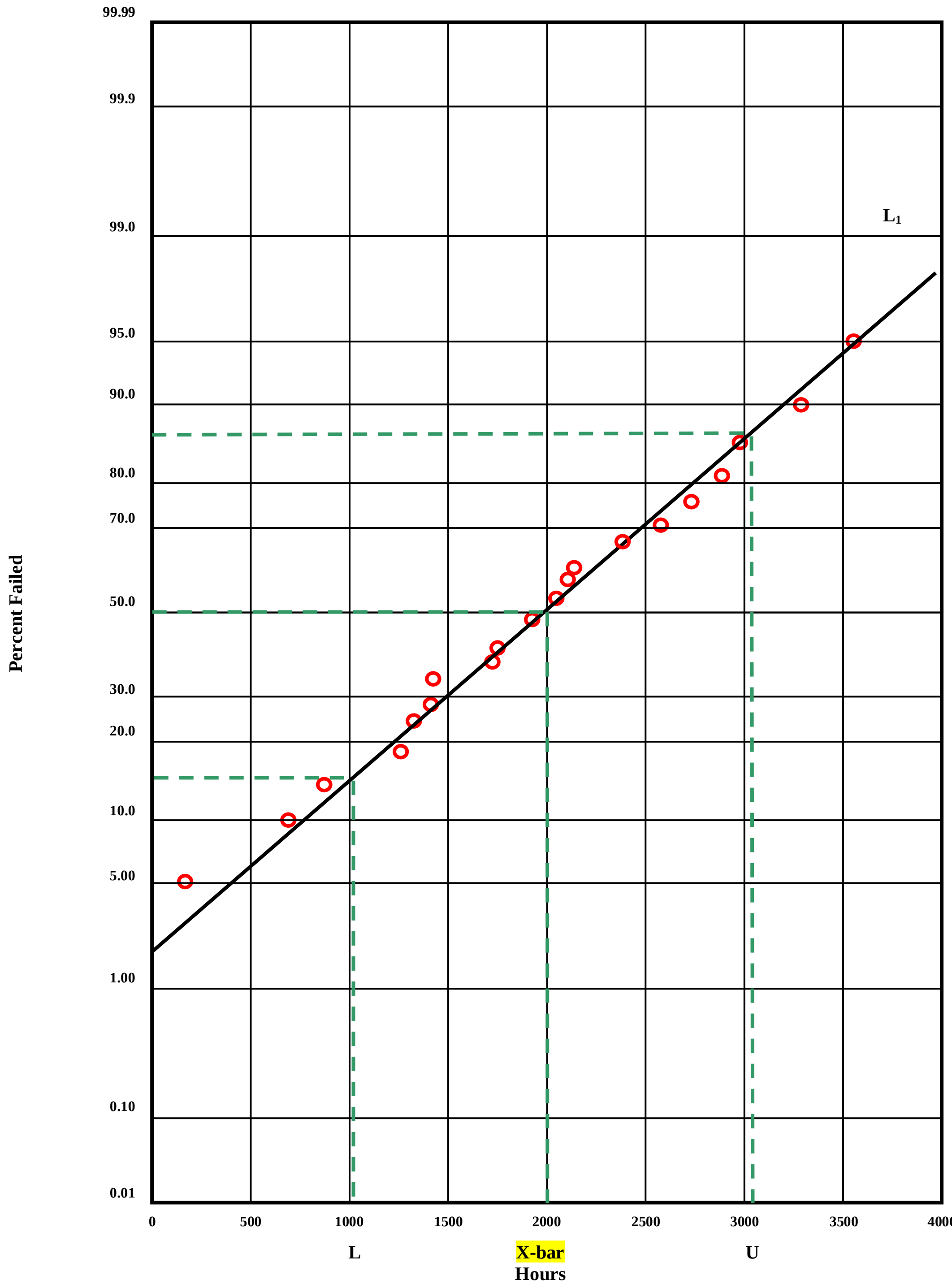


Figure 4.2-2: Graphical Point Estimation for the Normal Distribution

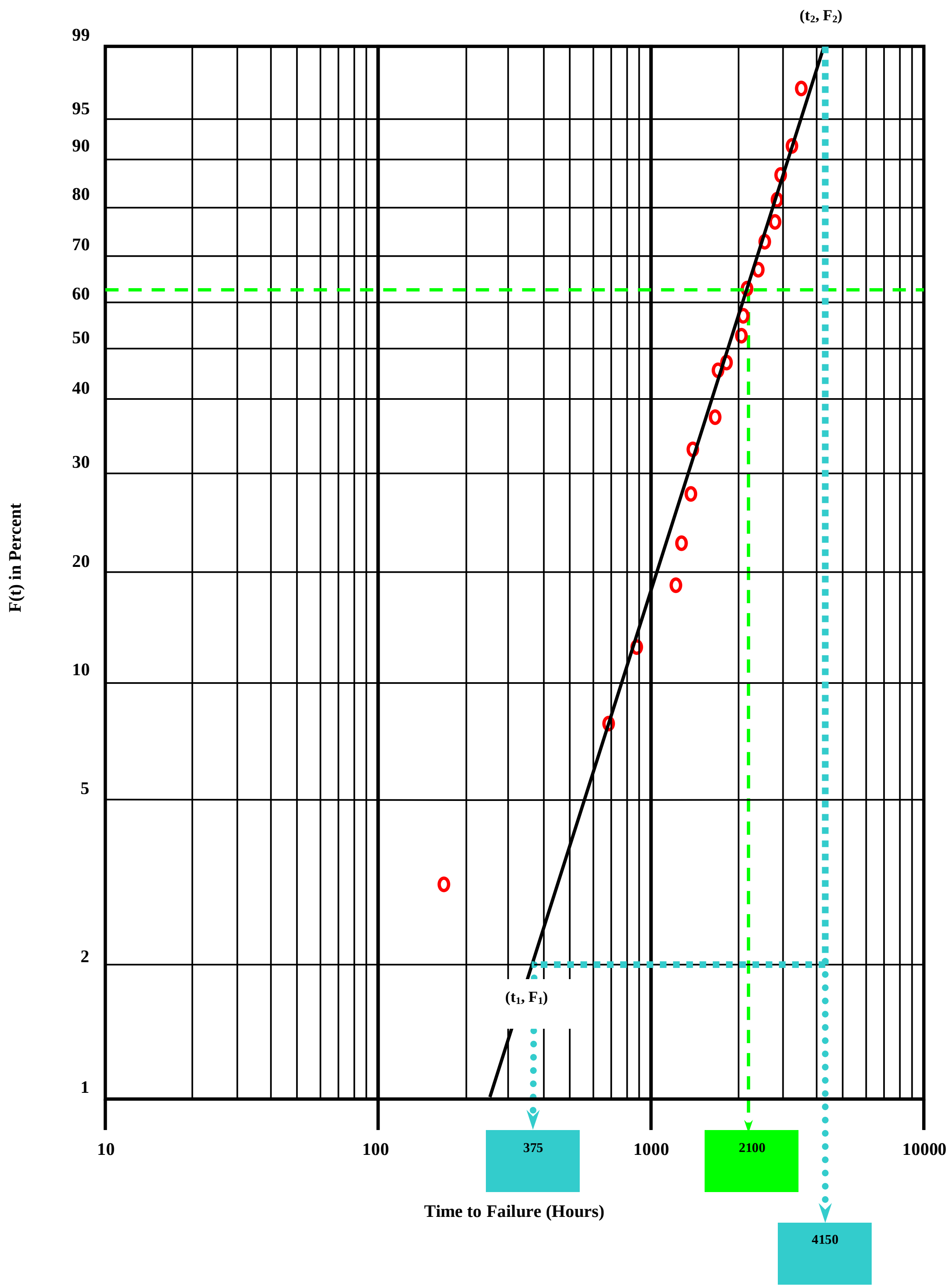


Figure 4.2-3: Graphical Point Estimation for the Weibull Distribution

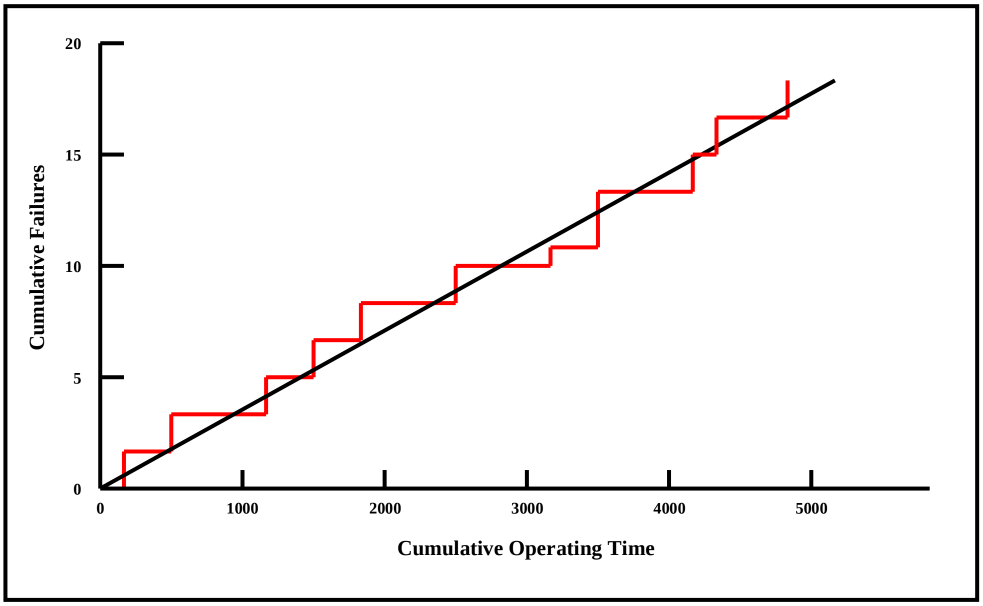


Figure 4.2-4: Graphical Evaluation of a Distribution

Reliability growth can be analyzed, either graphically or analytically, by using trend data. Graphical trend tests consist of plotting observed data such as the number of failures per unit time over time, or failure inter-arrival times in order to visually obtain the trend displayed by the data. Figure 4.2-7 illustrates the two failure time concepts, while Figure 4.2-8 provides an overview of a process for determining an appropriate reliability growth model type to use.

Table 4.2-5: Calculation of Reliability Parameters from Time Interval Data

Interval No./ Failure No.	Time Interval (t) (hour range)	No. of Hours in Interval	Fault Density f(t) x 10^{-2} hours	Hazard Rate h(t) x 10^{-2} hours	Cumulative Failures	Reliability R(t)
1	0 – 8	8	$\frac{1}{10*8}=1.25$	$\frac{1}{10*8}=1.25$	1	$\frac{10-1}{10}=0.90$
2	8 - 20	12	$\frac{1}{10*12}=0.83$	$\frac{1}{9*12}=0.93$	2	$\frac{10-2}{10}=0.80$
3	20 - 34	14	$\frac{1}{10*14}=0.71$	$\frac{1}{8*14}=0.89$	3	$\frac{10-3}{10}=0.70$
4	34 – 46	12	$\frac{1}{10*12}=0.83$	$\frac{1}{7*12}=1.19$	4	$\frac{10-4}{10}=0.60$
5	46 – 63	17	$\frac{1}{10*17}=0.59$	$\frac{1}{6*17}=0.98$	5	$\frac{10-5}{10}=0.50$
6	63 – 86	23	$\frac{1}{10*23}=0.43$	$\frac{1}{5*23}=0.87$	6	$\frac{10-6}{10}=0.40$
7	86 – 111	25	$\frac{1}{10*25}=0.40$	$\frac{1}{4*25}=1.00$	7	$\frac{10-7}{10}=0.30$
8	111 – 141	30	$\frac{1}{10*30}=0.33$	$\frac{1}{3*30}=1.11$	8	$\frac{10-8}{10}=0.20$
9	141 – 186	45	$\frac{1}{10*45}=0.22$	$\frac{1}{2*45}=1.11$	9	$\frac{10-9}{10}=0.10$
10	186 - 266	80	$\frac{1}{10*80}=0.13$	$\frac{1}{1*80}=1.25$	10	$\frac{10-10}{10}=0.00$

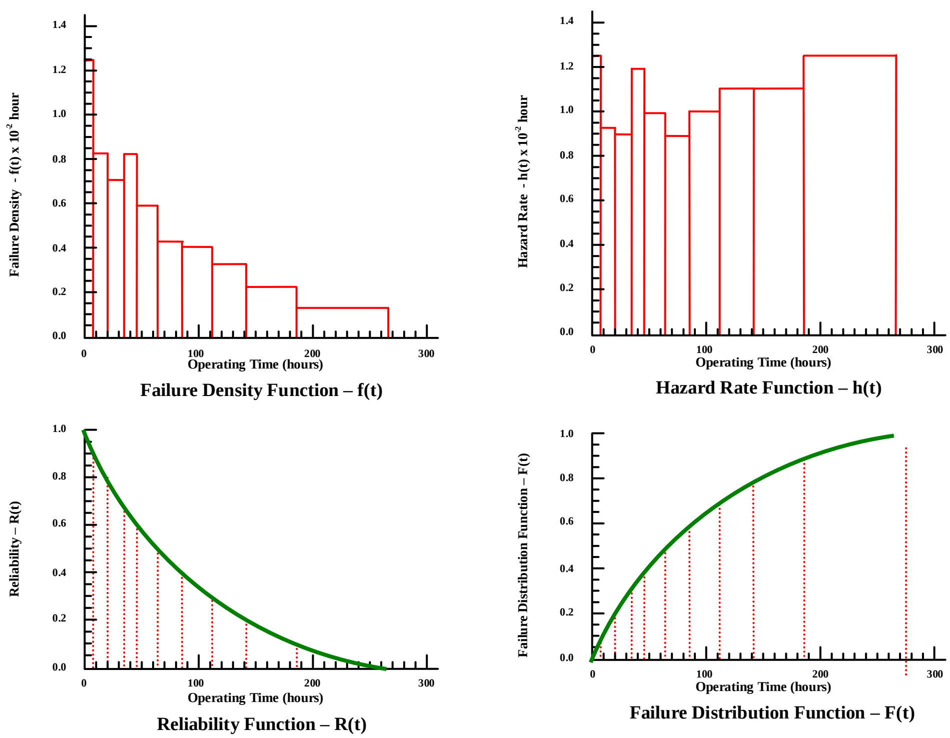


Figure 4.2-5: Reliability Parameters for Time Interval Data Example

Table 4.2-6: Calculation of Reliability Parameters from Failure Interval Data

Interval No.	Time to Failure (TTF) (hour range)	No. of Failures in Interval	Fault Density $f(t) \times 10^{-2}$ hours	Hazard Rate $h(t) \times 10^{-2}$ hours	Cumulative Failures	Reliability R(t)
1	0 – 2	222	$\frac{222}{1000*2}=11.10$	$\frac{222}{1000*2}=11.10$	222	$\frac{1000-222}{1000}=0.778$
2	2 - 4	45	$\frac{45}{1000*2}=2.25$	$\frac{45}{778*2}=2.89$	267	$\frac{1000-267}{1000}=0.733$
3	4 - 6	32	$\frac{32}{1000*2}=1.60$	$\frac{32}{733*2}=2.18$	299	$\frac{1000-299}{1000}=0.701$
4	6 – 8	27	$\frac{27}{1000*2}=1.35$	$\frac{27}{701*2}=1.92$	326	$\frac{1000-326}{1000}=0.674$
5	8 – 10	21	$\frac{21}{1000*2}=1.05$	$\frac{21}{674*2}=1.56$	347	$\frac{1000-347}{1000}=0.653$
6	10 – 12	15	$\frac{15}{1000*2}=0.75$	$\frac{15}{653*2}=1.13$	362	$\frac{1000-362}{1000}=0.638$
7	12 – 14	17	$\frac{17}{1000*2}=0.85$	$\frac{17}{638*2}=1.33$	379	$\frac{1000-379}{1000}=0.621$
8	14 – 16	7	$\frac{7}{1000*2}=0.35$	$\frac{7}{621*2}=0.56$	386	$\frac{1000-386}{1000}=0.614$
9	16 – 18	14	$\frac{14}{1000*2}=0.70$	$\frac{14}{614*2}=1.14$	400	$\frac{1000-400}{1000}=0.600$
10	18 - 20	9	$\frac{9}{1000*2}=0.45$	$\frac{9}{600*2}=0.75$	409	$\frac{1000-409}{1000}=0.591$
11	20 - 22	8	$\frac{8}{1000*2}=0.40$	$\frac{8}{591*2}=0.68$	417	$\frac{1000-417}{1000}=0.583$
12	22 - 24	3	$\frac{3}{1000*2}=0.15$	$\frac{3}{583*2}=0.26$	420	$\frac{1000-420}{1000}=0.580$
	TOTAL	**420**				

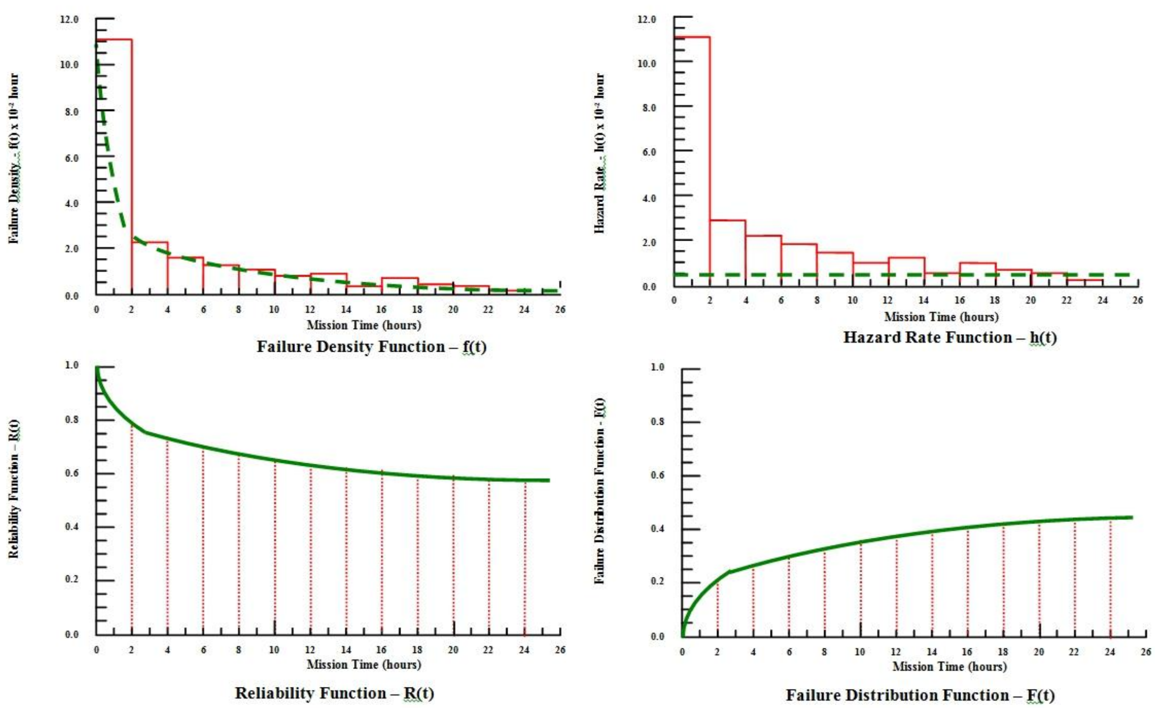
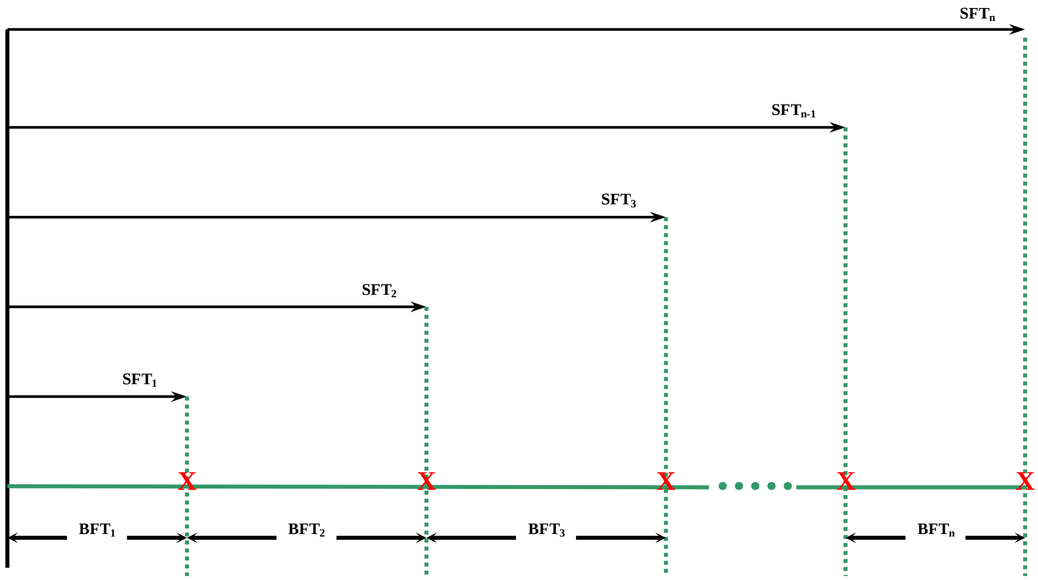


Figure 4.2-6: Reliability Parameters for Failure Interval Data Example



SFT$_i$ = System failure arrival times

The instance of occurrence of events measured from the time origin.

BFT$_i$ = Between failure arrival times

The measured time intervals between successive failure events.

System failure arrival times are the cumulative sum of all of the between-failure arrival times that preceded the current failure.

Figure 4.2-7: Determination of System Failure and Between Failure Arrival Times

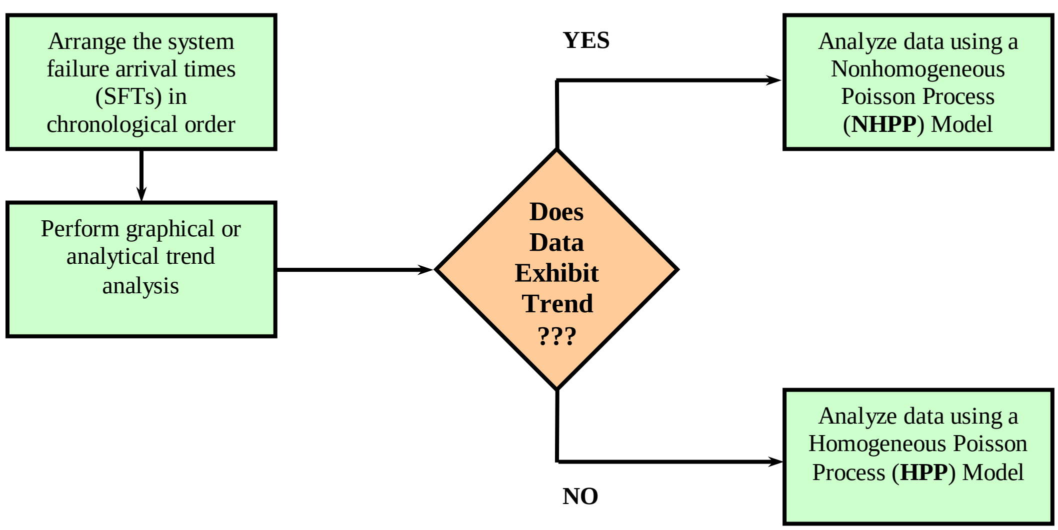


Figure 4.2-8: Determination of an Appropriate Process Model

Figure 4.2-9 provides a graphical illustration of data trend analysis, while Figure 4.2-10 shows the use of the Laplace statistic to draw conclusions about data trends.

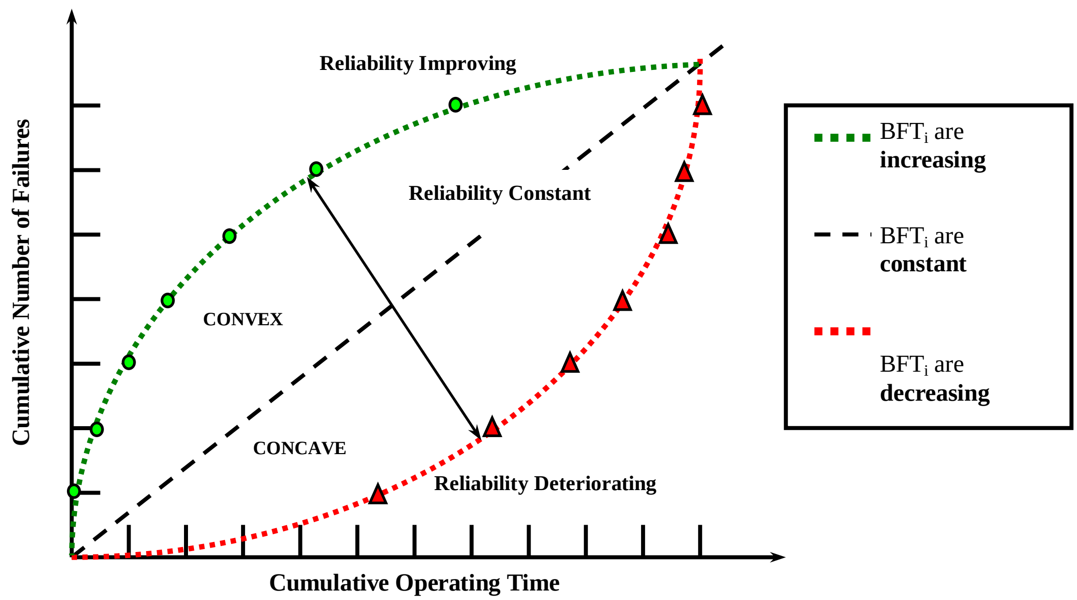


Figure 4.2-9: Graphical Representation of Failure Trends

The Laplace test statistic for a process with "n" failures is calculated using:

$$\mathbf{u} = \frac{\left[\left(\sum_{i=1}^{n-1} \mathbf{SFT_i}\right) \Big/ (\mathbf{n}-1)\right] - \left(\mathbf{SFT_n}/2\right)}{\mathbf{SFT_n}\sqrt{1/(12*(\mathbf{n}-1))}}$$

Trend analysis conclusions that can be drawn from the Laplace statistic are:

1. u $\cong 0$ (no apparent trend)
2. u > 0 (between failure intervals – BFT$_i$ – are tending to decrease, i.e., reliability growth is negative)
3. u < 0 (between failure intervals – BFT$_i$ – are tending to increase, i.e., reliability growth is positive)
4. When data is being plotted as failures occur, variability between $-2 < u < +2$ indicates that reliability is stable

Figure 4.2-10: Use of Laplace Statistic for Failure Process Trend Analysis

An example illustrating the calculation of the Laplace statistic follows. Table 4.2-7 contains data from 3 hypothetical systems (A, B and C), listing for each failure both the system failure arrival times and the between-failure arrival times. The summary of the calculation of the Laplace statistic based on these data is shown in Table 4.2-8.

Table 4.2-7: Sample System Failure Data

Failure Order Number (i)	System A		System B		System C	
	SFT_i	BFT_i	SFT_i	BFT_i	SFT_i	BFT_i
1	30	30	89	89	89	89
2	84	54	121	32	179	90
3	148	64	147	26	265	86
4	234	86	168	21	352	87
5	336	102	184	16	442	90
6	466	130	198	14	530	88
7	820	354	205	7	619	89

Table 4.2-8: Calculation of Laplace Statistic for Sample Systems

	System A	System B	System C
Given	n = total failures = 7 SFT_7 = 820 hours $\sum_{i=1}^{7} SFT_i$ = 1298 hours	n = total failures = 7 SFT_7 = 205 hours $\sum_{i=1}^{7} SFT_i$ = 907 hours	n = total failures = 7 SFT_7 = 619 hours $\sum_{i=1}^{7} SFT_i$ = 1857 hours
Calculate	$\mathbf{u} = \frac{(1298/6) - (820/2)}{820\sqrt{1/72}}$ $\mathbf{u} = -2.004$ The between failure times (BFT_i) for System A are increasing. Reliability growth is positive.	$\mathbf{u} = \frac{(907/6) - (205/2)}{205\sqrt{1/72}}$ $\mathbf{u} = +2.014$ The between failure times (BFT_i) for System B are decreasing. Reliability growth is negative.	$\mathbf{u} = \frac{(1857/6) - (619/2)}{619\sqrt{1/72}}$ $\mathbf{u} = 0.0$ The between failure times (BFT_i) for System C are relatively stable. System reliability is not changing.
Model	**Use an NHPP model to estimate/predict reliability**	**Use an NHPP model to estimate/predict reliability**	**Use a HPP model to estimate/predict reliability**

For More Information:

1. Fenton, N.E. and Pfleeger, S.L., "Software Metrics: A Rigorous and Practical Approach", International Thomson Publishing, May 1998, ISBN 0534954251

1. Grady, R.B., "Practical Software Metrics for Project Management and Process Improvement", Prentice-Hall, 1992, ISBN 0137203845

2. Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008

3. MIL-HDBK-189 "Reliability Growth Management"

4. "System Reliability Toolkit", Reliability Information Analysis Center, SRKIT, December 2005

Topic 4.2.1: Types and Sources of Reliability Data

Types of Data. The two major categories of reliability data are *development* (which includes all test data) and *field*.

Development Data. Development data include failure and repair/fix data and built-in test (BIT) effectiveness information, such as fault detection and fault isolation performance. Whenever failures occur during development or demonstration testing, the results of subsequent failure analysis, maintenance and corrective action activity should be documented.

In addition to the conditions of failure data, problems noted during troubleshooting are important to record. Tied to the failure information, such as failure mode and cause, such information helps evaluate the effectiveness of any diagnostic elements in correctly detecting and isolating a fault. If the fault was a false alarm detected by system BIT, this fact should also be recorded. If such a problem continues to exist, then an analysis should be required to determine why the problem exists and how it can be fixed.

All data should continuously be reviewed to determine if corrective actions are necessary to improve reliability. These reviews should be done in conjunction with and as part of a failure reporting, analysis and corrective action system, or FRACAS, which may or may not include a Failure Prevention Board, a Failure Review Board, or both. FRACAS is a closed-loop data reporting system for the purpose of systematically recording, analyzing, and resolving equipment reliability problems and failures.

To use FRACAS for data collection, appropriate data fields must be incorporated into a FRACAS data collection form. In addition to collecting data resulting from actual failure occurrences, information from simulations should also be documented.

Field Data. Field data include all operational information relevant to manual and automatic actions taken to operate an item in, or restore it to, an operable condition. These data include times to (or between) failure, environmental conditions and root failure cause and disposition (e.g., no fault found, relevant failure, independent failure, etc.). The information should also be classified according to when the failure or fault was discovered (i.e., preventive or corrective maintenance).

In designing a field reliability data collection system, or improving upon an existing system, it is important to minimize bias that can be introduced by those collecting the data. Therefore, keep in mind that operations and maintenance personnel should be trained on the data collection system, and its importance to tracking performance, identifying problems, and improving the product and product support characteristics.

In addition to reliability data being collected, other potential useful forms of data include customer or user satisfaction surveys. Such surveys should cover perceptions of system reliability performance and dependability.

Sources of Data. Reliability-related data may be obtained from several types of sources. Potential data sources include:

- Historical data from similar products
- Design or manufacturing data
- Data recorded during reliability testing
- Data provided by subcontractors and suppliers
- Field use data

The data may be expressed in a variety of terms. These include observed values or modified values (true, predicted, estimated, extrapolated, etc.) of the various reliability measures. Some precautions are therefore necessary regarding the understanding and use of such data as shown in Table 4.2.1-1.

Table 4.2.1-1: Sources of Data

<table>
<tr><th>Source</th><th>Comments</th></tr>
<tr><td>Historical</td><td>Used primarily during the concept definition phase to generate specification requirements. In the later phases, historical data may be compared with actual data obtained for the system, equipment or software. It can also serve as an additional source of information for reliability verification.

Before using, understand:
The origin of the data (e.g., field operation, in-house test or supplier-generated) and the system, equipment or software on which such data are based
Why and how the data apply to the current item
The methods used to collect the data, together with the training and skill levels of maintenance personnel involved (to ensure data quality and integrity)
Discrepancies that might affect the applicability of historical data to the product under consideration</td></tr>
<tr><td>Product Design and Manufacturing</td><td>Data obtained through the use of detailed design reliability analyses or assessments, or from data generated during the design phase or the manufacturing phase (e.g., accelerated life tests, reliability growth tests at the component level or higher, production reliability tests, etc.).

Design/manufacturing data may be used as the basis for:
Product qualification and acceptance (with regard to reliability requirements)
Review of the relevancy of historical data and the validity of previous reliability assessments
Before using, understand:
The data collection and analysis methodology used
Why the specific method was selected and applied
Any possible limitations in data accuracy</td></tr>
<tr><td>Product Demonstration and Field</td><td>These data are essential for sustaining engineering activities during the in-service phase of the item life cycle and include:
Reliability-related data obtained from formal or informal demonstration tests on mock-ups, prototypes or production equipment in either a true or a simulated environment
Data generated during actual item use (e.g., in-house test at the product-/system-level, field operations, etc.).
Before using, understand:
The methods for selecting specific actions, data monitoring and recording techniques
The skill level of maintenance personnel and the specific equipment training they have received (to ensure data quality and integrity)</td></tr>
</table>

Topic 4.2.2: Use of Existing Reliability Data

Development programs often make use of existing equipment designs or software code (i.e., software reuse), or designs/code adapted to a particular application. If this situation exists, the following table summarizes the necessary characteristics of the data needed for reliability analyses.

Table 4.2.2-1: Use of Existing Reliability Data

Information Required	**Field Data**	**Test Data**	**Component Data (HW or SW)**
Data collection time period	**X**	**X**	**X**
Number of operating hours/miles/cycles per equipment/system	**X**	**X**	
Total number of component hours/cycles/operations			**X**
Total number of observed corrective maintenance actions, or corrective maintenance actions required during preventive maintenance	**X**		
Number of "no defect found" maintenance actions (chargeable failures)	**X**		
Number of induced maintenance actions (non-chargeable failures)	**X**		
Number of "hard failure" maintenance actions (chargeable failures)	**X**		
Number of observed failures (total chargeable failures)	**X**	**X**	**X**
Number of relevant failures (analysis performed to root failure cause and based on Failure Definitions and Scoring Criteria)	**X**	**X**	**X**
Number of non-relevant failures (based on Failure Definitions and Scoring Criteria)	**X**	**X**	**X**
Failure definition (should be included in Specifications)	**X**	**X**	**X**
Number of systems, equipments or components to which data pertains	**X**	**X**	**X**
Similarity of system/equipment/component of interest to system/equipment/component for which data is to be used	**X**	**X**	**X**
Environmental stress and operating profiles associated with data	**X**	**X**	**X**
Type of testing		**X**	
Field data source	**X**		

Topic 4.2.3: Data Analysis Techniques

The precise form of analysis of data is specific to each use and the analysis can be complex and time-consuming. Experienced analysts who can properly assess the information to be extracted from the raw data should do the analysis.

Data are frequently analyzed to obtain statistical inferences regarding a given population of data. Statistical inference is the process of drawing conclusions about an entire population of similar objects, events, or tasks, based upon a sample of a few. Two basic approaches to statistical inference are mainly used:

***Parametric*:** This approach is primarily concerned with inference about certain summary measures of distributions (mean, variance, etc.). It is based on explicit assumptions about the population distributions and parameters.

***Non-parametric*:** This approach is concerned with inference about an entire probability distribution, free of any assumptions regarding the parameters of the population sampled.

Meaningful data handling and its subsequent evaluation also require some prior investigation of the process generating the data. Different sets of data available on an item may be combined, provided that the same selection criteria have been applied to each set. The choice of appropriate methods of data evaluation may be influenced by such factors as possible time-dependency of the process or more than one cause relating directly to the data.

Any peculiarities in the data collection scheme should be taken into account in analyzing the data. The analyst should identify any data falling outside a pre-set range. Acceptance or rejection criteria should be explicitly stated and validated.

Frequently, one of a number of types of statistical distributions will underlie the collected data. Three principal methods are available to identify a particular underlying distribution:

- Engineering judgment, based upon an analysis of the physical process generating the data
- Graphical methods using special charts, leading to the construction of nomographs
- Statistical tests, such as the Chi-square and goodness-of-fit, providing a measure of the deviations between the sample and the assumed distributions

Data Used Explicitly for Compliance Verification. When reliability-related data is to be used for compliance testing and for determination testing, the analysis procedures used need to be considered very carefully and discussed in detail in any subsequent test report. Table 4.2.3-1 summarizes some of the major areas of importance in using data for compliance verification.

Table 4.2.3-1: Areas to Consider in Using Data for Compliance Verification

Area	Comments
Data Editing/ Data Transposition	Describe the actions taken to ensure the accuracy, completeness and validity of the data. If any censoring is performed, present the rules and reasons for performing the censoring. If data are transposed from one form to another (e.g., from a linear to a logarithmic scale), clearly state the reason and justification for such a transposition.
Statistical Distribution Analysis	Usually necessary to determine the underlying distribution if the data are to be analyzed statistically. The most commonly used distribution functions in reliability are the exponential, Weibull, lognormal and Rayleigh (for many software-related datasets). Describe the method of testing the distribution assumption, with the reasons for that specific selection. Common methods used in reliability analysis include the χ^2 (chi-square), Kolmogorov-Smirnov (K-S) and various graphical tests. The K-S test (also known as *d*-test) is the most frequently used method for distribution testing.
Parameter Computation	Clearly state the basis for computing all reliability parameters to be presented. If selected parameters are to be computed on a cumulative or interval basis, detail the method to be used. Fully describe any reliability mathematical models to be used.
Presentation of Results	Clearly state all conditions needed for understanding and using the data. These conditions include the purpose of the data collection scheme, especially with respect to type and variation of the data chosen. Provide circumstantial information, such as time/date stamps, geographic locations and the calendar period over which the data was collected. Indicate particular situations that may limit the data application and use (for example, any difficulties encountered, assumptions, or incompleteness of data). Consider the best form of presentation. A condensed form (for example, diagrams, histograms, and graphical presentations) may be more appropriate than detailed numerical listings.

Three methods of analyzing data are outlined in this section. These methods are:

- Weibull Analysis
- Regression Analysis
- Analysis of Variance

Weibull Analysis

Waloddi Weibull developed the Weibull distribution in 1937 as a function that ". . . may sometimes render good service." The initial reaction to his paper on the new distribution, presented in America in 1951, was negative. Over the years, however, with improvements in plotting methods, rank ordering, and so forth, the Weibull has become the leading method for fitting life data.

Primarily a tool for solving reliability problems, the Weibull has wider applications, including maintainability. Some of the sample problems solvable using Weibull analyses are shown in Table 4.2.3-2.

Table 4.2.3-2: Problems Solvable Using Weibull Analysis

• How many components must be tested and for how long to verify reliability has been improved by 50% from the previous configuration
• A machine supplier claims that the failures occurring with his equipment are random events associated with operators. You think premature wear out is the cause. Who is right?
• You only afford to warranty 5% of your components. What must the scheduled replacement interval be?
• We have made design changes to correct previous problems. Are these changes working?
• How many spare parts must we keep on the shelf to maintain 95% availability?
• Eight failures of a component have occurred in the first year of service. How many will occur in the next 2 years?

Weibull analysis can be particularly helpful in a reliability-centered maintenance analysis. Specifically, Weibull analysis can tell the planner whether or not preventive maintenance (PM) is warranted. The value of the beta (β) parameter of the Weibull distribution indicates if the item under study is subject to wear-out. If it is not, then PM is not warranted. If it is, then PM should be planned <u>if</u> the cost of a failure is greater than the cost of the preventive maintenance. If PM is warranted, the Weibull analysis can be used to identify the optimum PM interval. Software upgrades can be planned to coincide with these optimum PM intervals.

Weibull analysis has many advantages over other methods of analyzing life data. It:

- Provides accurate results with few samples
- Renders simple and useful graphical results with the slope of the graph providing clues to physics of failure
- Can represent many distributions

Regression Analysis

An easy way to examine data is by a scatter plot. When we plot the points from the given set of data onto a rectangular coordinate system, we have a scatter plot. Regression analysis is a method for analyzing the relationship represented by the plot.

A regression equation is a mathematical equation that can be used to predict the values of one dependent variable from known values of one or more independent variables. The term is derived from the heredity studies performed by Sir Francis Galton in which he compared the heights of sons to the height of their fathers.

Linear regression is used to make predictions about a single value. Simple linear regression involves discovering the equation for a line that most nearly fits the given data. That linear equation is then used to predict values for the data. A regression analysis that involves only one predictor is called Simple Linear Regression Analysis. Even though a single predictor may oversimplify the estimation in real systems, the results that are obtained can be easily extended to real systems.

Linear regression involves a model of the form: $y = \beta_0 + \beta_1 x + \varepsilon$

This model is referred to as the linear model where y is the dependent variable, x is the independent variable, ε is experimental error (also called noise), and β_0 and β_1 are constants. The term linear refers to the coefficients. The highest power of x is termed the order of the model. A power of one denotes a first-order model. A second-order model would be of the form:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \varepsilon$$

A non-linear model is of the form: $\boldsymbol{y = \beta_0 + x^{\beta_1} + \varepsilon}$

Non-linear models are intrinsically difficult to solve, so we seek a suitable linear model or one that can be transformed to a linear model. An example of the latter is:

$$\boldsymbol{y = e^{\beta_0} + x^{\beta_1} + \varepsilon}$$

Taking the natural log of both sides of this equation transforms it into a linear equation.

$$\ln y = \beta_0 + \beta_1 \ln x + \ln \varepsilon$$

One method for estimating the parameters of a linear model is the least squares method.

Correlation describes the strength, or degree, of a linear relationship. That is, correlation lets us specify to what extent the two variables behave alike or vary together. Correlation analysis is used to assess the simultaneous variability of a collection of variables. Different methods are available for determining when the degree of correlation is statistically significant.

Analysis of Variance

Analysis of variance (ANOVA) is a technique for examining the influence of one or more nominal scaled independent variables on an interval- or ratio-scaled dependent variable in an experiment.

In many tests, it is necessary to compare the means of several populations simultaneously. In doing so, several important assumptions are made:

- The variation within each factor is the same
- The distributions of each population are Normal
- Errors are independent

In using ANOVA, the variations in test results (response measurement) are partitioned into components that reflect the effects of one or more independent variables. The variability of a set of measurements is proportional to the sum of the squares of deviations used to calculate the variance:

$$\text{Variability (Measurement Set)} = \sum \left(\boldsymbol{X} - \overline{\boldsymbol{X}} \right)^2$$

The sum of the squares of the deviations (total sum of squares) is partitioned into parts associated with the variables in the test plus a remainder that is associated with random error. When a test variable if highly related to the response, its part of the total sum of squares will be very large. An F-statistic test is used to confirm this by comparing the variable sum of squares with that of the random error.

One way in which ANOVA could be used for maintainability purposes is in determining if the mean time between failure for a software-intensive system varies from one operating location to another.

For More Information:

1. Abernethy, Dr. Robert B., "The New Weibull Handbook", Second Edition, Robert B. Abernethy, North Palm Beach, FL, 1996.
2. Focht, Stanley P., "Weibull Analysis Applications to Predictive Maintenance Programs", EPRI Conference on Predictive Maintenance, May 1994.
3. Hays, W.L. and Winkler, W.L. "Statistics-Probability, Inference and Decision", Holt, Reinhart and Winston, New York, NY, 1971.
4. IEC 60706-6, "Guide on Maintainability of Equipment - Part 6: Section 9: Statistical Methods in Maintainability Evaluation", 1994.
5. Glasser, Gerald J., "Planned Replacement: Some Theory and its Application", Journal of Quality Technology, Vol. 1, No. 2, April 1969.
6. Coppola, Anthony, "STAT: Practical Statistical Tools for the Reliability Engineer", Reliability Information Analysis Center, Rome, NY, 1999.
7. Knezevic, J., "Effective Analysis of Existing Maintainability Data", SAE Communications in RMS, Volume 2/Number 1, January 1995.
8. Mendenhall, William and Richard L. Scheaffer, "Mathematical Statistics with Application", Duxbury Press, North Scituate, MA, 1973.

Topic 4.2.3.1: Weibull Analysis

Weibull analysis continues to be popular for reliability work due to its inherent versatility. Many of the distributions used in reliability can be derived from or approximated by the Weibull density function. A sampling of the types of problems that can be solved through Weibull analysis includes:

- A machine supplier claims that failures occurring with their equipment are operator-related random events. You think premature wear-out is the cause. Who is right? (In Weibull analysis this reduces to a question of the value of β, the Weibull shape parameter).
- You can only afford to warranty 5% of your components. What should the scheduled replacement interval be? (This problem is solved by examining the Weibull plot of the data to determine the corresponding time to failure).
- Problems have been addressed with a design change. Does the design change correct the problem? (This reduces to examining the value of β for the failure mode addressed by the change).
- How many spare parts must be in the stockroom to maintain 95% availability? (This can be solved by examining the expected number of failures).
- During the first year of service, a product has failed 8 times. How many more failures are expected in the next 2 years? (This can be solved by examining the expected number of failures from the Weibull plot).

Table 4.2.3.1-1 illustrates several characteristics of Weibull analysis.

Table 4.2.3.1-1: Characteristics of Weibull Analysis

Advantages	Data Requirements	Plotting Procedures
• Accurate results with few samples • Provides simple/useful graphical results • Slope of graph provides physics of failure clues • Many distributions can be represented through Weibull analysis	• Requires "age" data • Life data that is relevant to the failure mode is critical • Examples of life data are number of cycles, miles, minutes, hours, operations, sessions, or start-ups to failure	• Order data from lowest to highest failure time • Estimate percent failing before each failure time (median ranks) • Draw best line fit through data points plotted on Weibull paper • Estimate Weibull parameters β and α from the graph

One needs life data to use Weibull analysis. Examples of life data are cycles, mileage, minutes, start-ups, operations (for software), hours, etc. The data can come from either field operation or testing, but the actual times-to-failure (life units) must be known. It is critical that the life data be relevant to the single prevalent failure mode in order to avoid ambiguous or misleading results in the interpretation of the data.

An advantage of using the Weibull analysis method is that simple graphical methods can be used to analyze the data. Data are plotted on special paper, called Weibull paper. This paper is unique in many ways, including the scales (ln-ln on the vertical axis and ln on the horizontal axis). The vertical axis represents the cumulative fraction of items, F(t), that will fail by a given time, t, and the horizontal axis represents the times-to-failure. A typical plot on Weibull paper is shown in Figure 4.2.3.1-1. (Note: this plot and the one in Figure 4.2.3.1-2 were drawn using a Weibull software package; however, manual plots on Weibull paper appear the same.)

From the Weibull equation, we can derive the following:

$$1 - F(t) = e^{-\left(t/\alpha\right)^{\beta}}$$

$$1/(1 - F(t)) = e^{-\left(t/\alpha\right)^{\beta}}$$

$$\ln\ln(1/(1 - F(t))) = \beta\ln(t) - \beta\ln(\alpha)$$

Since ln(t) is the scale of the horizontal axis and lnln(1/(1-F(t))) is scale of the vertical axis on Weibull paper, $y = \beta x + a$

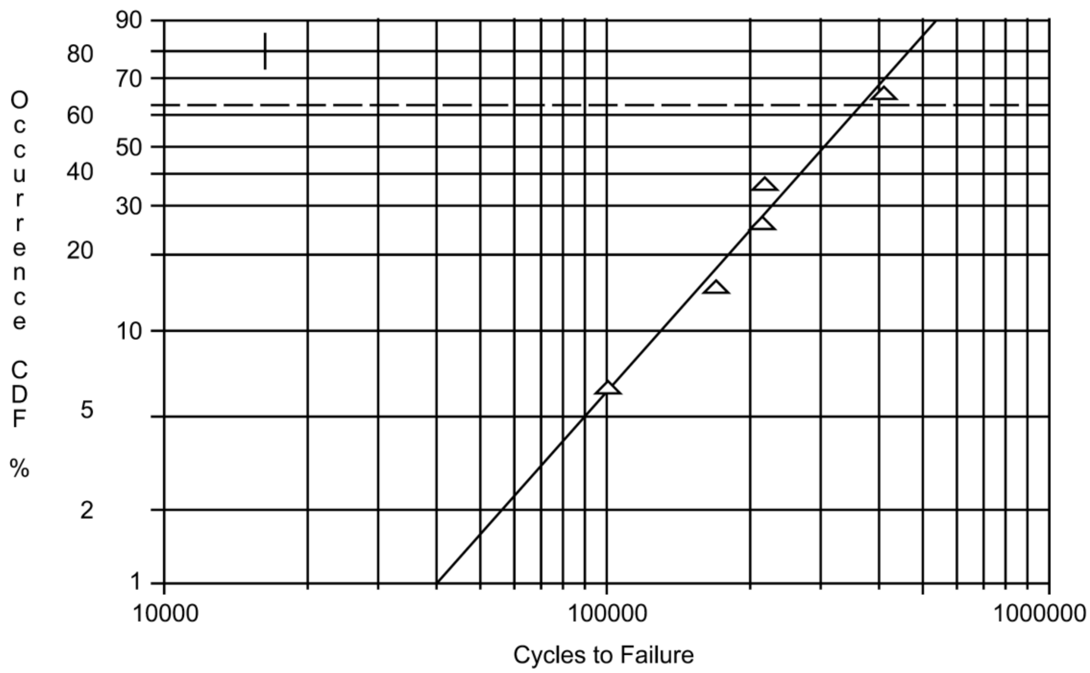


Figure 4.2.3.1-1. Typical Weibull Plot

Four steps are used to plot and analyze life data on Weibull paper.

1. Order the data from the shortest to the longest failure time
2. Estimate the percent of the population failing before each sample failure time (Median Ranks (MRs); see Table 4.2.3.1-1)
3. Draw a best-fit line through the data points
4. Estimate the Weibull parameters (beta,β, and alpha, α) from the graph

$\beta = \dfrac{\Delta Y}{\Delta X}$ of Weibull line and α = 63.2% percentile of F(t)

where: $Y = \ln\ln \dfrac{1}{(1 - F(t))}$ and $X = \ln(t)$

(Note: Some special Weibull graph paper allows β to be read directly. The characteristic life, α is the value on the x-axis found by dropping a vertical line from the point on the line corresponding to 63% cumulative probability of failure down to the x-axis. For any value on the x-axis, the value of F(t) can be found.)

Table 4.2.3.1-1: Median Rank Table

	SAMPLE SIZE (N)									
i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	.5000	.2929	.2063	.1591	.1294	.1091	.0943	.0830	.0741	.0670
2		.7071	.5000	.3864	.3147	.2655	.2295	.2021	.1806	.1632
3			.7937	.6136	.5000	.4218	.3648	.3213	.2871	.2594
4				.8409	.6853	.5782	.5000	.4404	.3935	.3557
5					.8706	.7345	.6352	.5596	.5000	.4519
6						.8909	.7705	.6787	.6065	.5481
7							.9057	.7979	.7129	.6443
8								.9170	.8194	.7406
9									.9259	.8368
10										.9330

For sample sizes greater than 10, and in a situation discussed later, Bernard's Approximation may be used rather than a median rank table. It is given by:

$$MR = \frac{\boldsymbol{i} - 0.3}{\boldsymbol{N} + 0.4} \text{ x } 100\%$$

where:

i = rank order
N = number tested

After the Weibull plot is complete, the result must be interpreted. Table 4.2.3.1-3 summarizes the ways in which the plot can be interpreted.

Table 4.2.3.1-3: Interpretation of a Weibull Plot

Slope (β)	Implies	Suspect
<1	**Infant mortality (decreasing failure rate)** If a component survives infant mortality, its resistance to failure improves with age	• Inadequate stress screening or burn-in • Quality problems in components or manufacturing, or both • Overhaul problems
= 1	**Failures are random (constant failure rate)** An old part is as good or bad as a new part. Scheduled replacement is not cost effective.	• Maintenance/human errors • Failures are "Acts of God" • Mixture of failure modes in complex parts or systems
>1 and <4	**Wearout (increasing failure rate)** Typical of most mechanical part failures. An old part is not as good as a new part. Scheduled replacement may be cost effective.	• Low cycle fatigue • Corrosion or erosion
>4	**Old age (end-of-life)** Old parts wear out (fail) rapidly.	• Problem with material properties • Brittleness (materials like ceramics) • Small variability in manufacturing or material

An example follows.

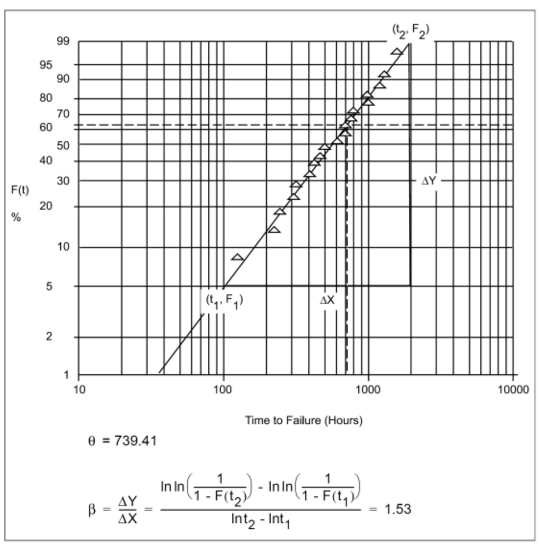
The following failure data are collected from a test in which 20 items were tested to failure.

Order Number	Failure Time (in hours)	Median Rank (%)	Order Number	Failure Time (in hours)	Median Rank (%)
1	92	3.41	11	640	52.45
2	120	8.31	12	700	57.36
3	233	13.22	13	710	62.26
4	260	18.12	14	770	67.17
5	320	23.02	15	830	72.07
6	325	27.93	16	1010	76.98
7	420	32.83	17	1020	81.88
8	430	37.74	18	1280	86.78
9	465	42.64	19	1330	91.69
10	518	47.55	20	1690	96.59

Figure 4.2.3.1-2 shows the data plotted on Weibull paper. From the graph, α is 739.41 hours. β is:

$$\beta = \frac{\Delta \boldsymbol{Y}}{\Delta \boldsymbol{X}} = \frac{\ln\ln\left(\frac{1}{1-0.99}\right) - \ln\ln\left(\frac{1}{1-0.05}\right)}{\ln 2000 - \ln 105} = 1.53$$

The reliability at t = 1000 hours is found by drawing a line up vertically from t = 1000 on the abscissa to the line. Then, from that point a horizontal line is drawn to the ordinate. It intersects the ordinate at F(t) = 80%. The reliability is 1- F(t) = 20% (i.e., 20% percent probability of no failure). Since β = 1.53 (>1, <4), the items exhibit wear-out. So scheduled replacement should be considered for the item. If the item were replaced every 100 hours, an average of only 5% will fail in service.



θ = 739.41

$$\beta = \frac{\Delta \boldsymbol{Y}}{\Delta \boldsymbol{X}} = \frac{\ln\ln\left(\frac{1}{1-\boldsymbol{F}(\boldsymbol{t}_2)}\right) - \ln\ln\left(\frac{1}{1-\boldsymbol{F}(\boldsymbol{t}_1)}\right)}{\ln \boldsymbol{t}_2 - \ln \boldsymbol{t}_1} = 1.53$$

Figure 4.2.3.1-2. Graphical Point Estimation for the Weibull Distribution

When not all items on test have failed, the times-to-failure data must be treated differently. The non-failures are called "suspensions". Since the tests are terminated before all items have failed, we call the suspensions "right suspensions." Right suspensions tend to increase α with little or no effect on β.

The plotting procedure for data with suspensions is:

- Rank all times, failures, and suspensions, earliest to latest
- Calculate the adjusted ranks for the failures (suspensions are not plotted) as follows:

$$\text{Adjusted Rank} = \frac{(\textbf{Inverse Rank})*(\textbf{Previous Adjusted Rank})+(N+1)}{(\textbf{Inverse Rank})+1}$$

- Apply Bernard's Approximation to calculate median ranks
- Plot failures versus median rank as before

An example using suspensions follows:

Eight gears are tested. Five fail and three are taken off test. The test times are:

Test Article	Test Hours	Result	Test Article	Test Hours	Result
1	110	F	5	2000	F
2	700	S	6	1460	S
3	600	F	7	6600	F
4	800	S	8	900	F

Where F = Failure and S = Suspension

We want to determine β and α, and determine what class of failure beta indicates. First, we rank all of the times.

Test Article	Test Hours	Result	Test Article	Test Hours	Result
1	110	F	5	900	F
2	600	F	6	1460	S
3	700	S	7	2000	F
4	800	S	8	6600	F

Where F = Failure and S = Suspension

Next, we calculate the inverted ranks (IR) and the adjusted ranks (AR) for the failures only and then calculate the median rank (MR).

RANK	IR	AR	RANK	IR	AR
1	8	1	5	4	3.4
2	7	2	6	3	-
3	6	-	7	2	5.3
4	5	-	8	1	7.2

Test Article	Test Hours	Result	AR	MR	Test Article	Test Hours	Result	AR	MR
1	110	F	1	8.33	5	900	F	3.4	36.90
2	600	F	2	20.24	6	1460	S	-	-
3	700	S	-	-	7	2000	F	5.3	59.52
4	800	S	-	-	8	6600	F	7.2	82.14

Where F = Failure and S = Suspension

Finally, we plot the times against the median ranks on Weibull paper. Doing so, we find that $\beta = 0.8$ and $\alpha = 3100$ hours. Since β is less than one, infant mortality is indicated.

Note: when a suspension and a failure occur simultaneously, place the failure first when ranking.

Sometimes, potting the data produces a curve with a sharp corner. In such cases,

- Two independent failure modes may be present
- The data points should be separated with non-included data points treated as suspensions
- Reliability is determined at any time as R(a)*R(b), where R(a) and R(b) are the results of the two plots of the separated data

Sometimes the plot simply is not straight. In those cases,

- Perhaps the Weibull distribution is not appropriate - try another distribution
- Maybe the origin is really not zero - try the three-parameter Weibull

The three-parameter Weibull is described by the following equation:

$$F(t) = 1 - e^{-\left(\left(t - t_0\right)/\alpha\right)^{\beta}}$$

where:

t_0 is the starting point or origin of the distribution
$t_0 > 0$ indicates a failure free period
$t_0 < 0$ indicates some life has been used up prior to testing

Before using the three-parameter Weibull, four criteria should be met:

- The Weibull plot should show a concave, downward curvature
- At least 20 failures should occur
- The correlation coefficient for the curve fit should significantly increase
- There should be a physical explanation why origin is not zero

Regarding a physical explanation why origin is not zero, some possible explanations are:

- Failure mode cannot happen instantaneously (some failure free time)
- Minimum stress level required for fracture
- Components deteriorate in storage (when first used, time is not zero)
- Burn-in was performed by the manufacturer

Although manually graphing life data on Weibull paper is a relatively easy and accurate method of analysis, it has largely been replaced by software-based tools. These include:

- *ReliaSoft's* Weibull++ - designed to perform Life Data Analyses as it applies to reliability engineering.
- *Fulton Finding's* WinSMITH - performs all of the Weibull techniques in Dr. Robert Abernethy's New Weibull Handbook, including likelihood ratio confidence, simplified design (set) comparison, Kaplan-Meier simulation and solution, critical correlation coefficient, and sudden-death Weibayes.
- *Relex Software Corporation's* WeibullSMITH - performs Weibull analysis of raw input data. Includes rank regression or maximum likelihood fitting, confidence bands, and three-parameter analysis.
- *Oliver Interactive, Inc.'s* RELCODE - a preventive maintenance tool for determining optimal replacement intervals for components. Using Weibull mathematics, RELCODE determines the probabilities of component-failure and helps the analyst decide whether to replace them at regular intervals, and if so the length of the interval, or only when a failure occurs.

Topic 4.2.3.2: Regression Analysis

Regression analysis is used to determine the relationship between variables. When the relationship is linear, we have linear regression analysis.

Correlation describes the strength, or degree, of a linear relationship. That is, correlation lets us specify to what extent the two variables behave alike or vary together. Correlation analysis is used to assess the simultaneous variability of a collection of variables. The relationships among variables in a correlation analysis are generally not directional.

As an example of correlation analysis, suppose one wants to study the simultaneous changes with age of height and weight for a population. Then, one can assess the height and weight changes in the population from infants to adults. Regression analysis describes how the change in height can influence the change in weight.

A popular method for estimating the parameters of a linear model is called least squares. It is a method for fitting a straight line to a set of data points. For example, suppose we want to fit a line having the form y = ax + b to a set of data pairs (x,y) shown in Figure 4.2.3.2-1. Fitting the line by eye is intuitive – we would try to keep the deviations of each data point "small." The least squares method is similar in that we minimize the sum of the squares of deviations (SSE).

$$\boldsymbol{SSE} = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} (\boldsymbol{y_i} - \boldsymbol{\hat{y}})^2 = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} [\boldsymbol{y_i} - (\boldsymbol{b} + \boldsymbol{ax_i})]^2$$

where $\boldsymbol{\hat{y}}$ is a point on the line and y_i is an observed point.

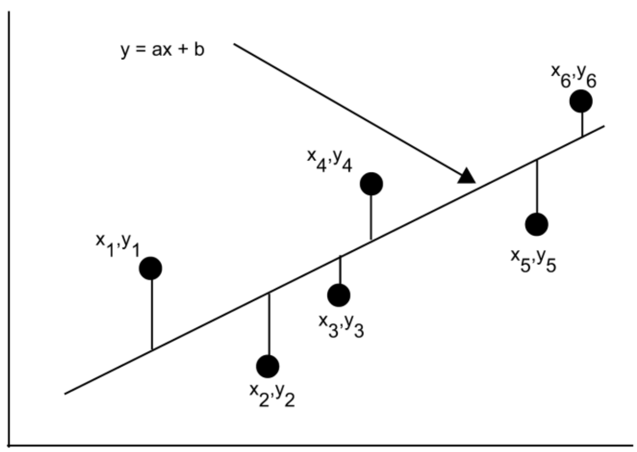


Figure 4.2.3.2-1. Fitting a Straight Line to a Set of Data Points

By taking the partial derivatives of the equation for SSE with respect to a and b and setting them equal to zero, we obtain the least-squares equations for estimating the parameters of a line. The equations are:

$$\frac{\partial SSE}{\partial b} = -2\left(\sum_{i=1}^{n} y_i - nb - a\sum_{i=1}^{n} x_i\right) = 0$$

$$\frac{\partial SSE}{\partial a} = -2\left(\sum_{i=1}^{n} x_i y_i - b\sum_{i=1}^{n} x_i - a\sum_{i=1}^{n} x_i^{\ 2}\right) = 0$$

Since both equations are linear, we can easily solve them simultaneously to obtain:

$$b = \bar{y} - a\bar{x}$$

We can determine "a" and "b" from the following equations:

$$a = \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i - \left[\left(\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i\right) \Big/ n\right]}{\sum_{i=1}^{n} x_i^{\ 2} - \left[\sum_{i=1}^{n} x_i \Big/ n\right]} \qquad b = \frac{\sum_{i=1}^{n} y_i - a\sum_{i=1}^{n} x_i}{n}$$

If the variables x and y are linearly related, then the correlation coefficient, r, is a measure of the degree of relationship present between the variables. The standardized correlation coefficient is defined as the covariance of x and y (covariance is a measure of the extent to which two random variables are related to one another) divided by the product of standard deviations of the x and y, and can be represented by the following form.

$$r = \frac{C_{xy}}{S_x S_y}$$

where,

$$C_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

$$S_x = \sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$S_y = \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

The correlation coefficient r varies from -1 to +1. A correlation coefficient of +1 indicates a perfect positive correlation; a value of zero indicates no correlation whatsoever, and a value of -1 indicates a perfect negative correlation.

As an example, assume the following data points:

x	1	2	3	4	5	6
y	1	2	3	4	5	6

Plotting the data yields the graph shown in Figure 4.2.3.2-2. From the graph, we observe that the slope of the line is +1. Since all points lie on the regression line, we have a perfect positive relationship between "x" and "y", and we can deduce that the sample correlation coefficient $\hat{r}$, is +1.00.

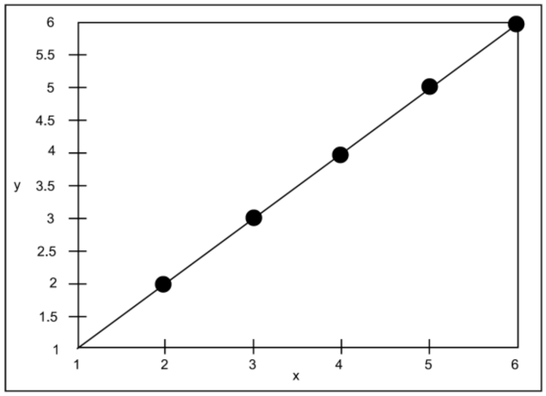


Figure 4.2.3.2-2. Plot of x and y

A key question is how large must the sample correlation coefficient be to indicate a significant correlation. Assuming that the data pairs have a bivariate normal distribution, testing for independence is equivalent to testing that the correlation coefficient, r, is zero (the null hypothesis).

The maximum likelihood estimator of r is given by the sample correlation coefficient:

$$\hat{r} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x - \bar{x})^2 \sum_{i=1}^{n}(y - \bar{y})^2}}$$

A first inclination would be to use $\hat{r}$ as the statistic for testing a hypothesis about r. Unfortunately, an exact derivation of this distribution is difficult. However, for moderately large samples, $\frac{1}{2}\ln\left[\frac{1+\hat{r}}{1-\hat{r}}\right]$ is approximately normally distributed, with mean $\frac{1}{2}\ln\left[\frac{1+r}{1-r}\right]$ and variance 1/(n-3). Thus, for testing the hypothesis that r = $\hat{r}$, we can use a z test in which:

$$z = \frac{(1/2)\ln\left(\frac{1+\hat{r}}{1-\hat{r}}\right) - (1/2)\ln\left(\frac{1+r}{1-r}\right)}{\left(\frac{1}{\sqrt{n-3}}\right)}$$

The null hypothesis will be rejected for $|z| > z_{\alpha/2}$, where α is the Type I error probability. Significance values of z are tabulated in standard tables such as Table 4.2.3.2-1.

Here is an example of how these tables are used. The following data on a number of similar systems at different geographic locations (represented by an ambient operating temperature in degrees centigrade) was obtained. The data concerned reliability performance (MTBF) and included recordings of other variables. Management needed to know, at a 95% confidence level, whether the observed reliability and the average system operating ambient temperature were correlated. The data were as shown in Table 4.2.3.2-2.

The data are plotted on a scatter diagram, as shown in Figure 4.2.3.2-3. Some negative correlation is indicated but cannot be confidently determined from the plot.

Table 4.2.3.2-1: Values of the Standard Normal Distribution Function

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.50000	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.52790	0.53188	0.53586
0.1	0.53983	0.54379	0.54776	0.55172	0.55567	0.55962	0.56356	0.56749	0.57142	0.57534
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62551	0.62930	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.65910	0.62276	0.66640	0.67003	0.67364	0.67724	0.68082	0.68438	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.70540	0.70884	0.71226	0.71566	0.71904	0.72240
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.75490
0.7	0.75803	0.76115	0.76424	0.76730	0.77035	0.77337	0.77637	0.77935	0.78230	0.78523
0.8	0.78814	0.79103	0.79389	0.79673	0.79954	0.80234	0.80510	0.80785	0.81057	0.81327
0.9	0.81594	0.81859	0.82121	0.82381	0.82639	0.82894	0.83147	0.83397	0.83646	0.83891
1.0	0.84134	0.84375	0.84613	0.84849	0.85083	0.85314	0.85543	0.85769	0.85993	0.86214
1.1	0.86433	0.86650	0.86864	0.87076	0.87285	0.87493	0.87697	0.87900	0.88100	0.88297
1.2	0.88493	0.88686	0.88877	0.89065	0.89251	0.89435	0.89616	0.89796	0.89973	0.90115
1.3	0.90320	0.90490	0.90658	0.90824	0.90988	0.91149	0.91308	0.91465	0.91621	0.91773
1.4	0.91924	0.92073	0.92219	0.92364	0.92506	0.92647	0.92785	0.92922	0.93056	0.93189
1.5	0.93319	0.93448	0.93574	0.93699	0.93822	0.93943	0.94062	0.94179	0.94295	0.94408
1.6	0.94520	0.94630	0.94738	0.94845	0.94950	0.95053	0.95154	0.95254	0.95352	0.95448
1.7	0.95543	0.95637	0.95728	0.95818	0.95907	0.95994	0.96080	0.96164	0.96246	0.96327
1.8	0.96407	0.96485	0.96562	0.96637	0.96711	0.96784	0.96856	0.96926	0.96995	0.97062
1.9	0.97128	0.97193	0.97257	0.97320	0.97381	0.97441	0.97500	0.97558	0.97615	0.97670
2.0	0.97725	0.97778	0.97831	0.97882	0.97932	0.97982	0.98030	0.98077	0.98124	0.98169
2.1	0.98214	0.98257	0.98300	0.98341	0.98382	0.98422	0.98461	0.98500	0.98537	0.98574
2.2	0.98610	0.98645	0.98679	0.98713	0.98745	0.98778	0.98809	0.98840	0.98870	0.98899
2.3	0.98928	0.98956	0.98983	0.99010	0.99036	0.99061	0.99086	0.99111	0.99134	0.99158
2.4	0.99180	0.99202	0.99224	0.99245	0.99266	0.99286	0.99305	0.99324	0.99343	0.99361
2.5	0.99379	0.99396	0.99413	0.99430	0.99446	0.99461	0.99477	0.99492	0.99506	0.99520
2.6	0.99534	0.99547	0.99560	0.99573	0.99585	0.99598	0.99609	0.99621	0.99632	0.99643
2.7	0.99653	0.99664	0.99674	0.99683	0.99693	0.99702	0.99711	0.99720	0.99728	0.99736
2.8	0.99744	0.99752	0.99760	0.99767	0.99774	0.99781	0.99788	0.99795	0.99801	0.99807
2.9	0.99813	0.99819	0.99825	0.99831	0.99836	0.99841	0.99846	0.99851	0.99856	0.99861
3.0	0.99865	0.99869	0.99874	0.99878	0.99882	0.99886	0.99889	0.99893	0.99897	0.99900

Table 4.2.3.2-2: System MTBF Data

System Number	MTBF	Ave. Operating Ambient Temp (°C)
1	7.88	28.0
2	7.01	30.7
3	4.97	9.7
4	4.74	18.1
5	6.34	18.2
6	4.59	28.1
7	11.39	12.2
8	10.11	14.1
9	8.18	9.6
10	8.32	16.7
11	7.74	16.1
12	7.00	15.8
13	9.39	7.1
14	9.28	8.5
15	10.93	14.2
16	1.11	30.9
17	8.18	13.5
18	7.68	15.7

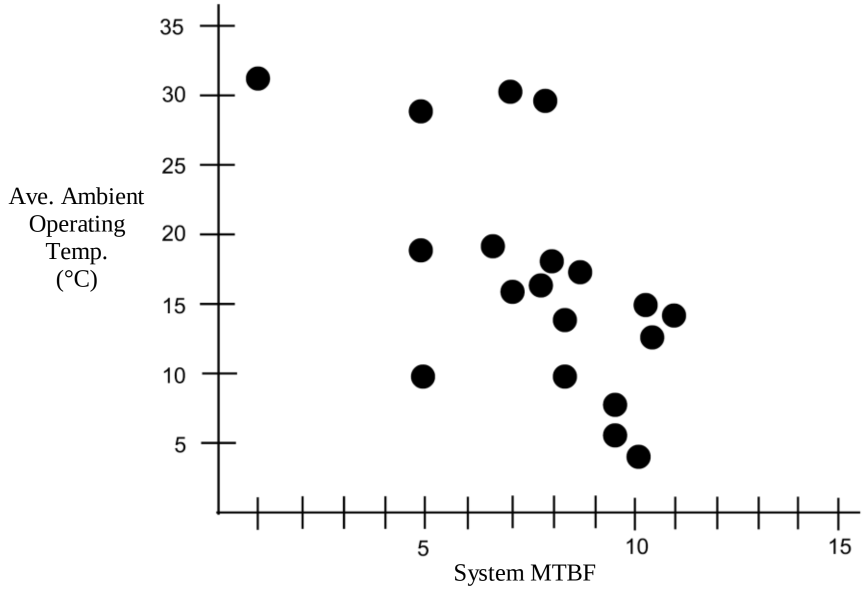


Figure 4.2.3.2-3. Scatter Plot of System MTBF and Average Operating Ambient Temperature

We assume that there is no relation between MTBF and the average operating ambient temperature. This assumption is called the null hypothesis. The alternative hypothesis is that there is a negative correlation. Since we are only interested in testing whether there is no correlation or a negative correlation (we are excluding a positive correlation), the test is called a one-tailed test.

To determine if there is any correlation, we first calculate the covariance of MTBF and operating temperature over the product of their standard deviations. Doing so yields:

$$\text{Cov (MTBF, temperature)} = -10.99$$

The MTBF and average ambient operating temperature standard deviations are estimated to be 2.5139 and 7.5208, respectively. Consequently, r is found to be:

$$\boldsymbol{r} = \frac{-10.99}{2.5139 * 7.5208} = -0.581$$

The test statistic is:

$$\boldsymbol{z} = \frac{(1/2)\ln\left(\frac{1+(-0.581)}{1-(-0.581)}\right) - 0}{\left(\frac{1}{\sqrt{15}}\right)} = \frac{-0.664}{0.258} = -2.574$$

From Table 4.2.3.2-1, the critical value, $z_{\alpha/2}$, is 1.96 where $\alpha = 0.05$ (1 - 0.95). Since the calculated absolute value exceeds the critical value, we reject the null hypothesis. The data strongly indicate a dependency between system MTBF and average ambient operating temperature.

Note that in calculating significance, we assumed that the variables MTBF and average ambient operating temperature were bivariate normally distributed. What if this is not the case? The Spearman rank correlation coefficient is a non-parametric means of measuring correlation that does not require the assumption of bivariate normally distributed variables.

To use Spearman's rank correlation, each observation is ranked. For example, for our systems, system #16 has the lowest reliability and would be ranked first with respect to MTBF. But it has the highest ambient operating temperature and would be ranked last with respect to that parameter. This dual ranking is done for each system. The Spearman's rank coefficient is given by:

$$\rho = 1 - \frac{6\sum_{j=1}^{m} d_j^{\,2}}{m^3 - m}$$

where:

m = the number of data pairs (in our example, 18)
d_j = the deviation between the two ranks for a given observation (in our example, each system)

We reject the null hypothesis of no dependency if the calculated statistic $\rho \leq -0.399$ (the value of -0.399 was obtained from a table of Critical Values of Spearman's Rank Correlation Coefficient). We calculate ρ, find that it is -0.637, and we reject the null hypothesis of independence.

Topic 4.2.3.3: Analysis of Variance

Analysis of variance (ANOVA) is a technique for examining the influence of one or more nominal scaled independent variables on an interval- or ratio-scaled dependent variable (in an experiment).

In many tests, it is necessary to compare the means of several populations simultaneously. In doing so, several important assumptions are made:

- The variation within each factor is the same
- The distributions of each population are normal
- Errors are independent

In using ANOVA, the variations in test results (response measurement) are partitioned into components that reflect the effects of one or more independent variables. The variability of a set of measurements is proportional to the sum of the squares of deviations used to calculate the variance:

$$\text{Variability (Measurement Set)} = \sum\left(X - \overline{X}\right)^2$$

The sum of the squares of the deviations (total sum of squares) is partitioned into parts associated with the variables in the test plus a remainder that is associated with random error. When a test variable is highly related to the response, its part of the total sum of squares will be very large. An F-statistic test is used to confirm the significance of the relationship by comparing the variable sum of squares with that of the random error.

Comparing Two Means: To compare the means of two different populations, the following formulas are used

Total Sum of Squares (Total SS) = SST + SSE

where:

SST = the sum of the squares between the two tests = $\dfrac{n_1 n_2}{n_1 + n_2}\left(\overline{X}_1 - \overline{X}_2\right)^2$

SSE = the sum of the squares within treatments (the error or residual term)

Therefore, SSE = Total SS - SST

The Total SS can be determined in two ways.

Total SS = (each observation - $\overline{X}$)^2
Total SS = S(each observation)^2 - CM (correction for the mean)

Two estimators are needed. These are:

$$\text{MST = Mean square of treatments} = \frac{SST}{Treatments - 1} = \frac{SST}{2-1} = SST$$

$$\text{MSE} = \frac{SST}{n_1 + n_2 - 2}$$

The test statistic for the null hypothesis, H_0: $\mu_1 = \mu_2$, is:

$$F = \frac{MST}{MSE} = \frac{\textbf{Mean variation between tests}}{\textbf{Mean variation within tests}}$$

The calculated test statistic is then compared with a critical value from an F-table (for a one-tailed F-test). The null hypothesis is rejected if the calculated statistic is larger than the critical F-value.

The following example illustrates the use of ANOVA in comparing two means.

In an experiment, six different users exercise two "identical" beta software programs (developed by different programmers) on a laptop computer. The software is tested to failure (defined as "the software crashes"). The results of the tests are shown in Table 4.2.3.3-1.

Table 4.2.3.3-1: Hours Until "Crash" for Two "Identical" Beta Software Programs

Software Program	Hours to Failure for Six Users	Total Hours	n	$\bar{X}$	Total Sum of Squares
A	5, 7, 9, 7, 6, 8	42	6	7	11.5
B	9, 10, 9, 5, 7, 8	48	6	8	17.5
TOTAL		90	12	7.5	29.0

The null hypothesis is that $\mu_A = \mu_{B.}$

$$\text{Total SS} = \sum\left(\text{each observation} - \bar{X}\right)^2$$

$$= \sum\ [(5 - 7.5)^2 + (7 - 7.5)^2 + \ldots + (8 - 7.5)^2]$$

$$= 29$$

$$\text{SST} = \frac{n_A n_B}{n_A + n_B}\left(\bar{X}_A - \bar{X}_B\right)^2 = 3(1)^2 = 3$$

SSE = Total SS - SST = 29 - 3 = 26

MST = SST = 3

$$\text{MSE} = \frac{26}{6+6-2} = 2.6$$

$$\text{F} = \frac{MST}{MSE} = \frac{3}{2.6} = 1.15$$

For a one-tailed F-test for comparing two means, with $\nu_1 = 1$ and $\nu_2 = n_A + n_B - 2 = 10$, at 95% confidence ($\alpha$ = 0.05), the value of F is 4.96. Since the calculated test statistic is smaller than the critical value of F, we cannot reject the null hypothesis.

Comparing More than Two Population Means: By extending the previous analysis, the ANOVA method can be used to detect differences among more than two population means. An explanation of how this is done follows.

Total SS = SST + SSE

Total SS = (SS of all values) - CM

CM is the correction for the mean

$$\text{CM} = \frac{(\textbf{Sum of all observations})^2}{n} = \frac{\left(\sum X\right)^2}{n}$$

$$\text{SST} = \text{test sum of squares} = \sum\frac{(\textbf{Sum of all observations})^2}{n} - CM$$

SSE = Total SS - SST

$$\text{MST} = \frac{SST}{t-1}$$

MSE = $\dfrac{SSE}{n-1}$

F = $\dfrac{MST}{MSE}$

If F > F_α, then the null hypothesis, $\mu_1 = \mu_2 = \mu_3 = . . . \mu_t$, is rejected.

An example follows.

The following tolerance measurements in material thickness were obtained from a single factor randomized experiment involving the output of three different machines.

Machine	Results (variation from nominal, in mm)					Average
A	3	0	-4	2	0	**0.2**
B	2	-2	1	0	5	**1.2**
C	2	1	-3	-3	1	**-0.4**

The question is, is there a statistical difference, at a 95% level of significance, in the performance of the three machines?

		Total of Observations	Total Observations	Sun of Squares of Observations
Test A	3, 0, -4, 2, 0	1	5	29
Test B	2, -2, 1, 0, 5	6	5	34
Test C	2, 1, -3, -3, 1	-2	5	24
	TOTAL	**5**	**15**	**87**

The number of observations, n, is 15. The number of tests, t, is 3. For the machines, the degrees of freedom are 2(t - 1). For the error, the degrees of freedom are 14(n - 1).

The critical F-statistic at a 95% level of significance and 2 and 12 degrees of freedom (the difference between the degrees of freedom for the error and the degrees of freedom for the machines) is:

$$F_{0.50_{2,12}} = 3.89$$

CM = $\dfrac{(\text{Sum of all observations})^2}{n} = \dfrac{\left(\sum X\right)^2}{n} = \dfrac{(5)^2}{15} = 1.67$

Total SS = (SS of all values) –CM = 87 – 1.67 = 85.33

SST = $\sum \dfrac{(\text{Sum of all observations})^2}{n} - CM = \dfrac{(1)^2}{5} + \dfrac{(6)^2}{5} + \dfrac{(-2)^2}{5} - 1.67 = 6.53$

MST = $\dfrac{SST}{t-1} = \dfrac{6.53}{3-1} = 3.27$

MSE = $\dfrac{SSE}{n-1} = \dfrac{78.8}{15-3} = 6.57$

F = $\dfrac{MST}{MSE} = \dfrac{3.27}{6.57} = 0.5$

Since the calculated value for F is less than the critical value of F, the null hypothesis cannot be rejected. In other words, there is no difference in the performance of the three machines at a 95% level of significance.

Appendix A: Reliability Basics

INTRODUCTION

The Handbook of Software Reliability and Security Testing is not intended to be a textbook on basic software reliability mathematical theory; however, a few of the basic principles and definitions need to be introduced so that the rest of the Handbook can be understood in context. Although the definition list may not be all inclusive, and the treatment of probabilistic subtleties may be somewhat less rigorous, there are many references included for those interested in a more in-depth discussion of reliability basics.

Appendix A.1: System Technical Performance Measures

Through the development of operational requirements for the system, specific *performance-related* factors are identified and applied with the objective of ensuring that the system will be designed to satisfactorily accomplish its mission. These factors, identified as *technical performance measures (TPMs),* may be applied as design-to criteria for the prime mission-related elements of the system and for those elements that are necessary to provide sustaining support of the system throughout its life cycle.

A number of metrics may be applicable in defining the design requirements for a system, and priorities need to be established in order to determine the relative degree of importance in the event that design trade-offs are necessary. For example:

- Is vehicle *speed* more important than *size*?
- Is *production quantity/capacity* more important than *product quality*?
- Is *performance range* more important than *reliability*?
- Is *computer memory capacity* more important than *processing speed*?
- Is *software usability* more important than *software functionality*?
- Is *packaging density* more important than providing *accessibility for diagnostics/maintenance*?

All of these considerations are important, and there will likely be a set of minimum requirements in each area. However, there may be a number of different design options, and systems engineers need to understand the priorities and interrelationships between customer/user needs and requirements. If compromises have to be made, *which requirements are more critical and where is additional emphasis required to arrive at an acceptable design solution?* The selected system configuration should reflect the necessary attributes or characteristics that are both responsive to the TPM requirements and consistent with the established priorities.

A basic approach for establishing specific design priorities is shown in Figure A.1-1. It is essential that good communications be established and maintained between the customer and the responsible design team. In early design review meetings, the TPMs derived from the system operational and support requirements should be reviewed and evaluated in terms of priorities. The most critical factors are identified and, thus, lead to areas where special design emphasis may be required. This first step (i.e., block 1, Figure A.1-1), representing the *voice of the customer (VOC),* identifies the WHATs, and the results may take a form such as shown in Table A.1-1. Referring to the example in the table, the most critical TPMs are Operational Availability (A_o), unit life-cycle cost (LCC), logistics response time, system velocity, and so on.

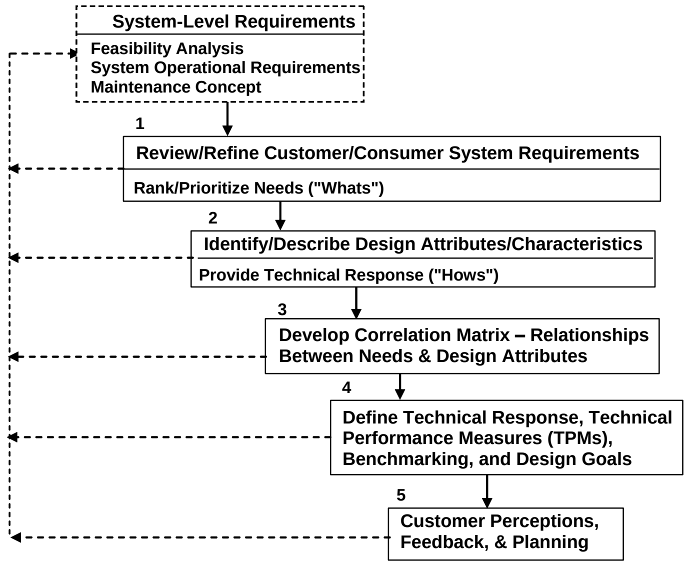


Figure A.1-1. Basic Steps in the Technical Performance Measure (TPM) Prioritization Process

Table A.1-1: Prioritization of Technical Performance Measures (TPMs)

TPM	Quantitative Requirement	Current Benchmark		Relative Importance (Customer "Needs")
		Metric	System	
Operational Availability (A_o)	98% (min)	98.5%	H	26%
LCC ($/unit)	$1.5M (max)	$3.3M	B	20%
Logistics Response Time (hrs)	2 hrs (max)	6 hrs	H	12%
Velocity (mph)	125mph (min)	100 mph	B	11%
Weight (lbs)	125K (max)	150K	H	9%
Size (ft)	Length: 125 ft Width: 12 ft Height: 10 ft (max)	Length: 136 ft Width: 15 ft Height: 12 ft	B	6%
MTBM (hrs)	300 hrs (min)	275 hrs	H	6%
Human Factors (error rate/yr)	< 1%	2%	D	5%
Information Process Time (hrs)	0.5 hrs (max)	2 hrs	B	5%
				100%

Referring to Figure A.1-1 (block 2), the next step is to identify the attributes, or characteristics, that need to be included and inherent within the selected system design configuration to comply with the requirements in Table A.1-1. By providing a good technical response, we begin to define the HOW requirements (in response to the WHATs). In other words, given that Operational Availability (A_o) is a key requirement, *what specific characteristics need to be built into the design in order to ensure that a 98% operational availability for the system will be attained?*

An excellent tool to aid in the establishment and prioritization of TPMs, as well as for the identification of appropriate technical responses, is quality function deployment (QFD). Implementation of QFD requires a team approach to ensure that the "voice of the customer" is reflected in the system design. The purpose is to establish the necessary requirements and to translate those requirements into technical solutions.

Customer requirements and preferences are defined, weighted based on their perceived degree of importance, and their attributes described. The QFD method provides the design team with an understanding of customer needs, focuses the customer on prioritizing those needs, and enables a comparison of competing design approaches. Each customer attribute is then satisfied by a technical solution.

The QFD process involves constructing one or more matrices, the first of which is often referred to as the *House of Quality (HOQ).* A modified version of the HOQ is presented in Figure A.1-2. Beginning on the left side of the structure, customer needs are identified and ranked in terms of priority, with levels of importance being quantified. This side reflects the "*WHATs*" that must be addressed. A team comprised of both customer and responsible design organizations determines the priorities through an iterative process of review, evaluation, revision, re-evaluation, etc. The top part of the HOQ identifies the designer's proposed *technical* responses relative to the attributes (characteristics) that must be incorporated into the design in order to respond to customer needs. This area constitutes the "*HOWs*". There should be at least one technical solution for each identified customer need. The interrelationships among attributes (or technical correlations) are identified, as well as possible areas of conflict. The center area of the HOQ conveys the strength or impact of the proposed technical response on the identified requirement. The bottom area allows for a comparison between possible alternatives. The information on the right side of the HOQ is used for planning purposes.

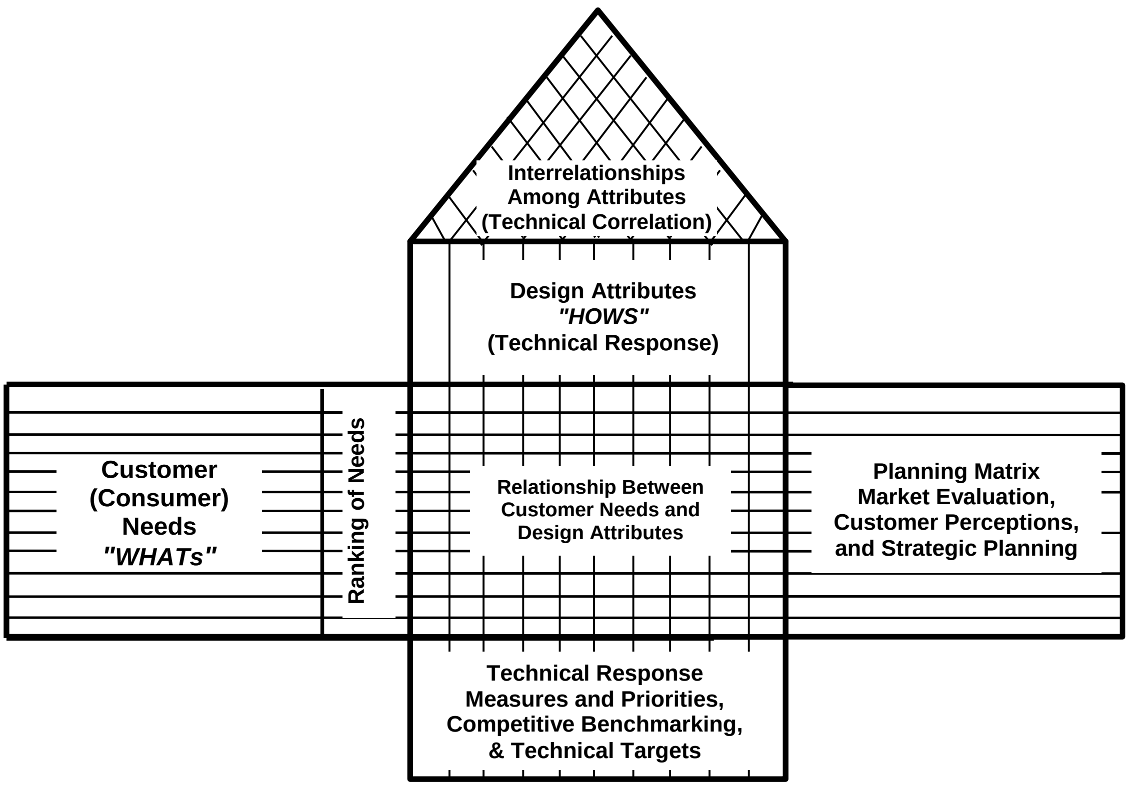


Figure A.1-2. Modified House of Quality (HOQ)

The QFD method facilitates the translation of a prioritized set of subjective customer requirements into a set of system-level requirements during conceptual design (i.e., the Concept Refinement Phase). A similar approach may be used to subsequently translate system-level requirements into a more detailed set of requirements at each stage in the design and development process. In Figure A.1-3, the *HOWs* from one house become the *WHATs* for a succeeding house. Requirements may be developed for the system, subsystems, components, manufacturing process, etc. The objective is to ensure the required justification and traceability of requirements from the top down to the lowest defined level of indenture.

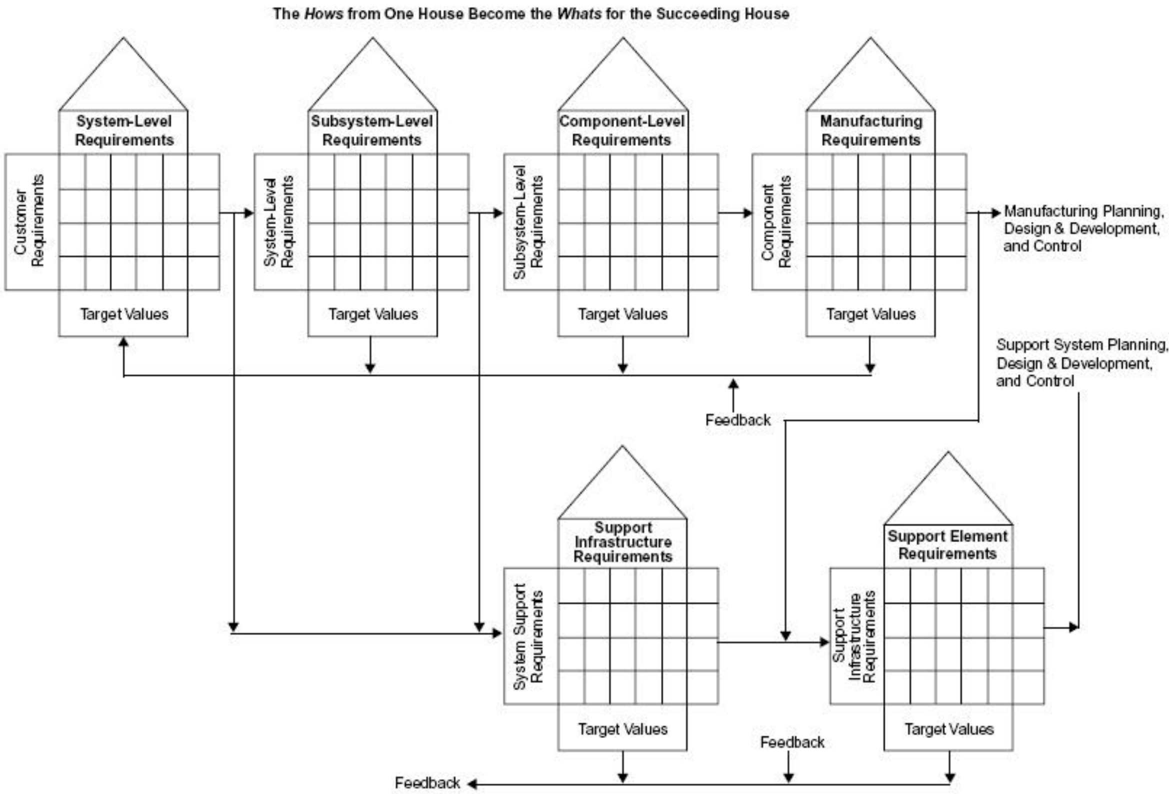


Figure A.1-3. The QFD Family of Houses

For More Information:

1. Blanchard, B.S. and Langford, J.W., "Supportability Toolkit", Reliability Information Analysis Center, Feb. 2005

Appendix A.2: Software and System Reliability Definitions

Table A.2-1: Basic Definitions of Common Software and System Reliability Terms

Term	Definition
Acceleration Factor	A factor by which a software operation is more frequently executed in test than it would be in the field due to its criticality.
Assertion	Software code that checks the value of a variable against an anticipated value or range of values and generates a return code or message if the check is not valid.
Attribute	A characteristic of a software operation, represented by a node or set of nodes in a graphical representation of an operational file.
Availability	A measure of the degree to which an item is in an operable state at any time.
Benchmarking	Rating a company's practices, designs, processes against the world's best practices for purposes of seeking improvement.
Bottleneck Processor	A processor that requires the most execution time per natural or operating time unit.
Build	A minor release of software that incorporates defect fixes and possible new features; multiple builds occur as a major release is developed.
Built-in Test (BIT)	An integral capability designed into a product which provides an automated test capability to detect or isolate failures.
Certification Test	A test that is directed solely at accepting or rejecting the software and is not coupled to removal of the faults that are causing the failures.
Computer Utilization	The ratio of execution time to total time.
Confidence Limit	One extreme (upper or lower) of a range in which a specified percentage of the true value of a variable occurs.
Consumer Risk (β)	Used in conjunction with statistical testing. The probability of a customer accepting an item (the objective is falsely met) which would be proven bad (the objective is really not met) if the test was conducted for an infinite time (or population).
Control Charts	Statistical charts derived from measuring factory processes. Used to spot process "drift" and inherent process variations which designers must account for in the basic design to achieve a "robust design."
Criticality	The importance of an operation with respect to safety or value added by satisfactory execution, or risk to human life, cost or system capability resulting from failure.
Curve Fitting	The use of statistical regression analysis to study the relationship between software complexity and the number of faults in a program, as well as the number of changes, or the failure rate
Defect Density	The number of defects per thousand lines of code (KLOC) or function points. Defect density depends on (a) the software development process, (2) software complexity, (3) experience of software development team, (4) percentage of code reused from previous stable projects and (5) the level of testing before it is shipped.
Derating	Using an item in a way that applied stresses are below rated values.
Developed Code	New or modified executable delivered instructions.
Direct Input Variable	A variable external to software operation that controls execution of operation in an easily traceable way, allowing recognition of the relationship between values of the variable and the processing that results. This provides information to optimally select values to test.
Discrimination Ratio	For software, a factor of error in estimating the failure intensity in a software certification test.
Environmental Stress Screening (ESS)	A series of tests conducted under environmental stress often greater than experienced in normal operation to disclose weak parts and workmanship defects to be corrected.
Equivalence Class	A set of levels of direct input variables that yield the same failure behavior in a system because they cause identical processing, provided that execution involves identical sets of levels of indirect input variables.

Table A.2-1: Basic Definitions of Common Software and System Reliability Terms (continued)

Term	Definition
Error	An incorrect or missing action by a person(s) that causes a fault in a software program.
Error Seeding	An estimation of the number of errors in a software program using multistage sampling..
Estimation	The determination of software reliability model parameters and quantities from failure data.
Execution Time	The time that a processor(s) is/are executing non-filler operations, measured in execution hours.
Fail Set	The set of runs (and, hence, input states) that will cause a software fault to generate a failure.
Failure Intensity	Failures per natural or time unit
Failure Intensity Objective (FIO)	The failure intensity that a system is expected to meet prior to release to the field.
Failure Intensity Reduction Objective (FIRO)	The failure intensity improvement that must be obtained through software reliability strategies.
Failure Prevention Goal	The proportion of failures that fault tolerant features must prevent
Failure Rate (λ)	The total number of failures within an item population, divided by the total time expended by that population, during a particular measurement interval under stated conditions.
Fault	A system defect that causes a failure when the software is executed. A software fault is considered a defect in the software code.
Fault Density	The number of software faults per line of deliverable executable source code or per function point.
Fault Detection	A process which discovers the existence of faults. Can be accomplished manually or automatically, depending on product requirements.
Fault Exposure Ratio	A proportionality factor that relates failure intensity to the rate at which faults would be encountered if the software program were executed linearly.
Fault Isolation	The process of determining the location of a fault to the extent necessary to effect repair. Can be accomplished manually or automatically, depending on product requirements.
Fault Reduction Factor	The ratio of faults removed to failures experienced.
Fault Tolerance Fraction (FTF)	The part of the remaining failure intensity reduction objective (FIRO), after early system test and reviews, that is to be achieved through fault tolerance, as contrasted to system test.
Feature Test	A test that executes all the new test cases of a release, independently of each other, with interactions and effects of the field environment minimized, in order to identify failures resulting from test cases by themselves.
FI/FIO	The ratio of failure intensity to failure intensity objective, used to track status during testing.
Hazard Rate	Instantaneous failure rate. At any point in the life of an item, the incremental change in the number of failures per associated incremental change in time.
Homogeneity	The fraction of a set of test runs that exhibit the same failure behavior.
Homogeneous	Exhibiting the same failure behavior.
Indirect Input Variable	A variable external to software operation that influences execution of operation in a way that is not easily traceable, making it impractical to recognize the relationship between values of the variable and the processing that results and, therefore, to optimally select values to test.
Initial Failure Intensity	The failure intensity at the start of test, usually system test.
Input Space	The set of all possible input states for a software program.
Input State	The complete set of input variables for a test run, and their values.

Table A.2-1: Basic Definitions of Common Software and System Reliability Terms (continued)

Term	Definition
Load Test	A test that executes all test cases together, with full interactions and all of the effects of the field environment, whose purpose is to identify failures resulting from interactions among test cases, overloading of and queuing for resources, and data degradation.
Mean-Time-Between-Failure (MTBF)	A basic measure of reliability for repairable items. The average time during which all parts of the item perform within their specified limits, during a particular measurement period under stated conditions.
Mean-Time-Between-Maintenance (MTBM)	A basic measure of reliability for repairable fielded systems. The average time between all system maintenance actions. Maintenance actions may be for repair or preventive purposes.
Mean-Time-Between-Critical-Failure (MTBCF)	A measure of system reliability which includes the effects of any fault tolerance which may exist. The average time between failures which cause a loss of a system function defined as "critical" by the customer.
Mean-Time-To-Failure (MTTF)	A basic measure of reliability for non-repairable systems. Average failure free operating time, during a particular measurement period under stated conditions.
Module Usage Table	A list of the modules of a software program, with the probabilities that each is used on any given run of the program.
Natural Unit	A unit other than time related to the amount of processing performed by a software-based item, such as runs, pages of output, transactions, jobs, queries, etc.
Nonfiller	Refers to software operations other than fillers, and the natural or time units that they use.
Nonhomogeneity	The fraction of a set of test runs that exhibit the dissimilar failure behavior.
Nonhomogeneous	Exhibiting dissimilar failure behavior.
Occurrence Probability	The probability with which a software operation or attribute value occurs.
Occurrence Proportion	The proportion of occurrences of a new operation with respect to occurrences of all new operations for a software release.
Occurrence Rate	The frequency at which a software operation or attribute value occurs.
Operating Time	Hardware: The elapsed time from when an equipment is energized and performing at some level of functionality, until such time when the equipment is fully de-energized (dormant, with no functionality) Software: The elapsed time from the start to the end of program execution, to include those periods when the processor(s) is/are idle, but energized.
Operation	A major system logical task performed for the initiator, which returns control to the system when the operation is complete.
Operation Interaction Factor	A factor that estimates the effect of newly-developed software operations on reused software operations in causing failures, with typical values ranging from 0.1 to 0.25.
Operational Architecture	The structure of, and relations between, software operations as they are invoked in the field.
Operational Development	Software development that is scheduled operation by operation, in such a fashion that the most used and/or most critical operations are implemented in the first release, and the less used and/or less critical are delayed. The net result is faster time to market for the most used/critical capabilities.
Operational Profile	The complete set of operations (major system logical tasks), with their probabilities of occurrence.
Prediction	For software, the determination of software reliability model parameters and values derived from the software product and development process.
Producer Risk (α)	Used in conjunction with statistical testing. The probability of a customer rejecting an item (the objective is falsely not met) which would be proven good (the objective really is met) if a test was conducted for an infinite time (or population). Synonymous with Supplier Risk.
Program	For software, a set of complete instructions (operators, with operands specified) that executes within a single computer and relates to the accomplishment of some major function.

Table A.2-1: Basic Definitions of Common Software and System Reliability Terms (continued)

Term	Definition
Quality Function Deployment	A system that focuses on exactly what the customer wants. Activities which don't contribute to customer goals are considered wasteful and are eliminated.
Projection	An estimate for a failure point in the future.
Reduced Operation Software (ROS)	Software that directly implements only the most used and/or most critical operation(s), handling the other operations in some alternative fashion; the software analog for RISC for hardware.
Redundancy	The existence of one or more means (not necessarily identical) for accomplishing a given function. Active redundancy has all items operating simultaneously, while standby redundancy has alternate means activated upon failure.
Reliability	The probability that an item will perform its intended function for a specified interval under stated conditions.
Reliability Growth	The change in reliability (assumed positive; negative growth is possible) over the total life cycle, as a function of time (or the number of software test cases). Positive growth results from the successful identification and mitigation of deficiencies.
Reuse	For software, an operation (or operations) that has (have) been carried over from a previous software release and used, as is, in a new software release.
Robust Design	A design approach that accounts for limitations in production capabilities, such as accounting for production machinery tolerance limitations.
Run	The specific execution of a software operation, characterized by a complete set of input variables, with their associated values.
Soak Time	The amount of time since the last data reinitialization.
Software Reliability Strategy	An activity that reduces failure intensity, incurring development cost and, potentially, development time.
Stable Program	A software program in which the code is unchanging, with the program neither evolving nor having any faults removed.
Supplier Risk	Used in conjunction with statistical testing. The probability of a customer rejecting an item (the objective is falsely not met) which would be proven good (the objective really is met) if a test was conducted for an infinite time (or population). Synonymous with Producer Risk.
Test Case	The partial specification of a software run, characterized by a complete set of direct input variables, with their associated values.
Test Operational Profile	A modified operational profile that will be used to direct the test controller in executing a load test.
Test Procedure	For software, a test controller for a load test that invokes test cases at various times that are randomly selected from the test case set. Selection from the test case set is based on the test operational profile. Invocation times are based on the total operation occurrence rate.
Testability	A design characteristic which allows the status of the unit to be confidently determined in a timely manner.
Unit	For software, a part of a software system that is usually developed by one programmer and does not necessarily perform a complete function for either a user or another system.

For More Information:

1. Musa, J.D., "Software Reliability Engineering: More Reliable Software Faster and Cheaper", AuthorHouse, ISBN 1-4184-9387-2 (sc), August 2004
2. Bazovsky, I., "Reliability Theory and Practice," Prentice-Hall, 1961.
3. O'Connor, P., "Practical Reliability Engineering," Wiley, 1991.
4. Birolrni, A., "Quality and Reliability of Technical Systems," Springer-Verlay, 1994.

Appendix A.3: Software Reliability Figures-of-Merit

This section highlights metrics that can be directly measured from actual test or field experience at either the software or hardware component level, or at the system level.

The basic elements associated with system reliability metrics relate to faults/failures over (or at) some period of time, although metrics do exist that quantify reliability as a function of non-time bases, such as the number of software program runs, or the number of cycles or miles accumulated (mechanical reliability). The three primary time elements that are used in operational system reliability, maintainability and availability are:

Execution time: The actual CPU time spent by a computer in executing software (for software reliability). This can also be defined as the amount of time for human response following receipt of an external stimulus (for human reliability)

Calendar time: The real-time experience of people, expressed as days, weeks, months, years, etc. (for hardware, software and human factors reliability)

Clock time: The elapsed time, from start to end, of system operation (periods during which the system is shut down do not count) (hardware, software and human factors reliability)

Failures, as well as time, can be expressed in a variety of different ways, from the perspective of reliability, maintainability and availability:

Cumulative failure function: Defines the average cumulative failures associated with each point in time (also called the **mean value function**)

Failure intensity function: Represents the rate of change of the cumulative failure function (can be increasing, decreasing or constant over a given time period, depending on the software failure trend)

Failure rate function: Defines the rate per unit time that a failure will occur over a defined time period (e.g., calendar hour, operating hour, CPU execution hour, etc.)

$$\lambda = \frac{\text{Number of inherent failures experienced}}{\text{Total time period over which inherent failures were experienced}}$$

If 15 inherent failures are experienced over 2000 software execution hours, then the failure rate of the software is 15/2000, or 0.0075 failures per execution hour.

If relevant data can be collected, other environment-dependent maintenance measures that may prove beneficial are:

- Ratio of total defect repair time to total number of defects repaired (for software)
- Number of unresolved problems (e.g., CNDs)
- Time spent on unresolved problems
- Percentage of design changes or enhancements that introduce new faults (defects)
- Number of hardware or software modules required to be modified in order to incorporate an effective change

Relevant reliability figures-of-merit that are generally used at the system or equipment level include the following:

Mean time to failure (MTTF): Represents the average expected time from the occurrence of a one failure to the occurrence of the next failure (traditionally applied to non-repairable systems)

MTTF includes only inherent failures within a system. Actions resulting from scheduled preventive maintenance, or from induced and can-not-duplicate (CND) incidents are not counted towards MTTF.

If a non-repairable hardware component accumulates 500,000 operating hours, experiencing 18 inherent failures over that time span, then the mean time to failure is 500,000/18, or 27,778 operating hours.

Mean time between failure (MTBF): Represents the average expected time from the occurrence of one failure to the occurrence of the next failure (traditionally applied to repairable systems)

MTBF includes only inherent failures within a system. Actions resulting from scheduled preventive maintenance, or from induced and can-not-duplicate (CND) incidents, are not counted towards MTBF. If only failures that are critical to system performance or mission success are assessed, then mean time between critical failure (MTBCF) becomes an appropriate metric.

The MTBF can be calculated in a manner similar to MTTR, or it can be calculated from the reciprocal of the failure rate ($1/\lambda$). If the failure rate of a system is measured as 0.0075 failures per software execution hour, then the system MTBF is 1/0.0075, or 133.33 software execution hours.

Reliability Function: Quantifies the probability that an item will perform its intended function for a specified interval under stated conditions. In the case of systems and software, the exponential distribution is considered to be appropriate for determining item reliability.

$$\mathbf{R} = \mathbf{e}^{-\lambda \mathbf{t}} \text{ or } \mathbf{R} = \mathbf{e}^{-\mathbf{t}/\mathbf{MTBF}}$$

where,

R = Probability of successful performance over time period "t"

t = Time period of interest (in time units consistent with MTBF or λ)

λ = Measured, predicted or estimated failure rate of the item

MTBF = $1/\lambda$ = Measured, predicted or estimated mean time between failure of the item

If the measured failure rate of an item is 0.0000375 failures per operating hour, then the reliability of the item over a period of 1 year (8760 operating hours, assuming 24/7 operation with no downtime) is calculated as:

$$\mathbf{R} = \mathbf{e}^{-(0.0000375)(8760)} = \mathbf{e}^{-0.3285} = 0.72$$

The reliability metrics defined above are predicated on either time-based failure data (time of failure; time interval between failures) or failure-based failure data (cumulative failures up to a specified time; failures experienced during a time interval), each of which is illustrated in Figure A.3-1.

Software-Specific Reliability Metrics [Reference 6]: The referenced article by Dr. Norman Schneidewind suggests a number of software reliability figures-of-merit adapted from the updated IEEE 982.1 "Standard Dictionary of the Software Aspects of Dependability" and other references. These are summarized in this section.

Time Between Failures Trend: If the trend is increasing, positive reliability growth is suggested. If the trend is decreasing, negative reliability growth is suggested.

$$M_{i+1} = \left(T_{i+2} - T_{i+1}\right) > M_i = \left(T_{i+1} - T_i\right)$$

where,

M_ι = Trend of a series of time between failures

T_i = Time between failures

Trend Analysis: Indicates whether a trend in time between failures indicates positive or negative reliability growth.

$$U_i = \sum_{i=1}^{N_i} \frac{M_i - \dfrac{N_i T_i}{2}}{T_i \sqrt{N_i / 12}}$$

where,

U_t	=	Reliability growth trend
N_i	=	Actual cumulative number of failures at interval "*i*"
T_i	=	Time during which the N_i failures occur
M_i	=	Series of time being evaluated

Predicted software reliability:

$$R(T_i) = e^{-\left\{(\alpha/\beta)\left[e^{-\beta(T_{i-s+1})} - e^{-\beta(T_{i-s+2})}\right]\right\}}$$

where,

α	=	Initial failure rate
β	=	Rate of change of failure rate
T_i	=	Time for which the prediction is made
s	=	The first time interval when failure data is used in estimation of parameters α and β

Actual software reliability:

$$R_a(T_i) = 1 - \frac{x_i}{X_t}$$

where,

x_i	=	Number of failures observed in interval "*i*"
X_t	=	Total cumulative number of failures observed at interval "*t*"

Reliability Required to Meet Mission Duration:

$$R(T_s + T_m) = e^{-\left\{(\alpha/\beta)\left[e^{-\beta(T_{t-s+1})} - e^{-\beta(T_{t-s+2})}\right]\right\}}$$

where,

T_s	=	Mission start time (nominally, the last test time)
T_m	=	Mission time
T_t	=	$T_s + T_m$

Rate of Change of Software Reliability:

$$\frac{dR(T_i)}{dT_i} = \alpha R(T_i)\left[e^{-\beta(T_{t-s+1})} - e^{-\beta(T_{t-s+2})}\right]$$

Parameter Ratio (PR): Ranks the reliability of a set of software modules or releases before extended reliability prediction efforts. As PR becomes more positive, software reliability increases. The assumption, of course, is that β represents a positive rate of change in the failure rate (i.e., failure rate decreases). Ineffective software reliability engineering processes that introduce more software faults when they attempt to correct one can result in a negative rate of change (i.e., a negative PR).

$$PR = \frac{\beta}{\alpha}$$

Software Restoration Time:

$$T_i = \left(\frac{-1}{\beta}\right) Ln \left[\frac{-\beta \ln\left(R(T_i)\right)}{\alpha\left(1 - e^{-\beta}\right)}\right] + (s+1)$$

where,

T_i	=	Restoration Time
$R(T_i)$	=	Required reliability once the system has been restored

Predicted Cumulative Failures: The function $F(T_i)$ will increase at a decreasing rate if positive reliability growth is present.

$$F(T_i) = \left(\frac{\alpha}{\beta}\right)\left[1 - e^{-\beta\left(T_{i-s+1}\right)}\right] + X_{s-1}$$

where,

$F(T_i)$	=	Predicted cumulative failures at time "T_i"
T_i	=	The time when $F(T_i)$ failures are predicted to occur
X_{s-1}	=	Observed failure count in the range $\{s\text{-}1, T_i]$

Fault Correction Rate and Delay: The assumption is that the rate of fault correction is proportional to the rate of failure detection, i.e. it "keeps up" with the failure detection rate, except for a delay in correcting a fault. If, in practice, this assumption is not valid, the metric will underestimate the remaining faults in the software code.

Fault Correction Rate:

$$c_i = \frac{x_i}{T_{i+1} - T_i}$$

where,

c_i	=	Fault correction rate for fault "*i*"
x_i	=	The actual number of faults corrected in interval "*i*"

Mean Fault Correction Rate:

$$m_i = \sum_{i=1}^{i} \frac{x_i}{\left(T_{i+1} - T_i\right) n_{ci}}$$

where,

m_i	=	Mean fault correction rate in interval "*i*"
n_{ci}	=	The predicted number of faults corrected in interval "*i*"

Cumulative Probability Distribution of the Fault Correction Delay: Due to potentially large variability in fault correction times, the emphasis is on predicting limits, as opposed to expected values. This metric is intended to place an upper bound on the fault correction delay time.

$$F\left(delayT_i\right)=1-e^{-m_i\left(delayT_i\right)}$$

where,

F($delayT_i$) = Cumulative probability distribution of the fault correction delay, "*delayT*$_i$"(see below)

Upper Limit of the Fault Correction Delay: Used to compute the limit of $delayT_i$ using the specified limit for F($delayT_i$), above

$$delayT_i=\frac{\left\{-\ln\left[1-F\left(delayT_i\right)\right]\right\}}{m_i}$$

where,

$delayT_i$ = The fault correction delay for the mean fault correction rate of interval "*i*"

Predicted Cumulative Number of Faults Corrected: Assumes that the times between failure are equal to the times between faults.

$$N_{ci}=\sum_{i=1}^{i}\left[c_i\left(T_{i+1}-T_i\right)\right]$$

where,

N_{ci} = Predicted cumulative number of faults corrected at interval "*i*"

T_i = Time between failures

Proportion of Faults Corrected: Assumes that the number of faults equals the number of failures.

$$P_{ci}=\frac{N_{ci}}{N_i}$$

where,

N_{ci} = Predicted cumulative number of faults corrected at interval "*i*"

N_i = Cumulative number of actual failures observed at interval "*i*"

Predicted Failure Rate: The derivative of the Predicted Cumulative Failures.

$$f_t=\frac{dF\left(T_i\right)}{dT_i}=\alpha e^{e^{-\beta\left(T_{i-s+1}\right)}}$$

where,

f(T_i) = Predicted failure rate

Predicted Number of Failures in Interval "*i*":

$$m(T_i) = \frac{\alpha}{\beta}\left[e^{-\beta(T_{i-s+1})} - e^{-\beta(T_{i+1-s+1})} \right]$$

where,

$m(T_i)$ = Predicted number of failures in interval "*i*"

Predicted Normalized Number of Failures in Interval "*i*":

$$M(T_i) = \frac{m(T_i)}{S}$$

where,

$M(T_i)$ = Predicted normalized number of failures in interval "*i*"

S = Size of the software program, in thousand lines of code (KLOC)

Predicted Maximum Number of Failures Over the Software Life (at $T_i = \infty$): To ensure that this prediction is conservative, infinity is used as the software life.

$$F(\infty) = \frac{\alpha}{\beta} + X_{s-1}$$

where,

$F(\infty)$ = Predicted maximum number of failures at infinity

X_{s-1} = Observed failure count in the range $\{s\text{-}1, T_i = \infty]$

Predicted Maximum Number of Remaining Failures Over the Software Life (at $T_i = \infty$): Indicator of residual faults and failures that remain after testing is completed.

$$RF(t) = \frac{\alpha}{\beta} + X_{s-1} - X_t$$

where,

$RF(t)$ = Predicted maximum number of remaining failures after test time "*t*"

X_1 = Cumulative number of failures observed at the last test time "*t*"

Predicted Operational Reliability/Quality: Indicates, on a fractional (percentage) basis, the extent of fault and failure removal.

$$Q(t) = 1 - \left[\frac{RF(t)}{F(\infty)}\right]$$

where,

$Q(t)$ = Predicted operational quality/reliability

Probability of x_i Failures: Provides a measure of risk of operating the software, based on the Poisson process.

$$p(x_i) = \left[\frac{(m_i)^{x_i}}{(x_i)!}\right] e^{-m_i}$$

where,

$p(x_i)$ = Probability of x_i failures occurring during interval "*i*"

m_i = Mean number of cumulative failures in interval "*i*", computed as:

$$m_i = \frac{X_i}{\sum_{i=1}^{i} X_i}$$

where,

X_i = Cumulative number of failures occurring in interval "*i*"

Predicted Number of Faults Remaining to be Corrected: This metric can be calculated once the "maximum number of failures over the life of the software" and the "cumulative number of faults corrected" have been predicted. Assumes number of faults is equal to the number of failures.

$$R_{ci} = F(\infty) - N_{ci}$$

where,

R_{ci} = Predicted number of faults remaining to be corrected in interval "*i*"

Predicted Fault Correction Quality:

$$Q_{ci} = 1 - \left(\frac{R_{ci}}{F(\infty)}\right)$$

where,

Q_{ci} = Predicted fault correction quality in interval "*i*"

Weighted Failure Severity (for a Software Release): The higher the value of this metric, the lower the quality of the software release. See Table A.3-1 for example definitions of the failure severity codes

$$W_r = \sum_{i=1}^{N} \left(\frac{X_i}{N}\right)\left[1 - \left(\frac{S_{i-1}}{S_m}\right)\right]$$

where,

W_r = Weighted failure severity for a software release "*r*"

S_i = Severity of the fault "*i*" (the lower the value, the more severe the fault. See Table 3.3-1)

S_m = Maximum value of s_i (i.e., the minimum severity)

X_i = Number of failures of severity s_i

N = Number of failures that occurred on software release "*r*"

Table A.3-1: Example Definitions of Failure Severity Codes

Failure Severity Code	Potential Definition of Code
S_1	Loss of life or system
S_2	Impacts ability to complete mission objectives (including degraded operation)
S_3	Workaround available, therefore minimal effects on procedures. Mission objectives met.
S_4	Insignificant violation of requirements or recommended practices. Not visible to user during operational use
S_5	Cosmetic issue which should be addressed or tracked for future action, but not necessarily a current problem.

Software Metrics Modified from IEEE 982.1:

Actual Mean Time to Failure (MTTF):

$$MTTF_{actual}(T_i) = \left(\frac{1}{N_i}\right)\left[\sum_{i=1}^{N_i}(T_{i+1} - T_i)\right]$$

where,

N_i = Number of cumulative failures at failure "*i*"

Predicted Mean Time to Failure (MTTF): Reliability growth is demonstrated by an increasing $MTTF_{actual}(T_i)$ *and* $MTTF_{predicted}(T_i)$*, as a function of test time (or field time)* T_i

$$MTTF_{predicted}(T_i) = \left[\sum_{i=1}^{N_i}\frac{(T_{i+1} - T_i)}{F(T_i)}\right]$$

where,

$F(T_i)$ = Predicted cumulative number of failures at time "T_i"

Actual Failure Rate: The form of this metric is designed to demonstrate reliability growth, if it exists.

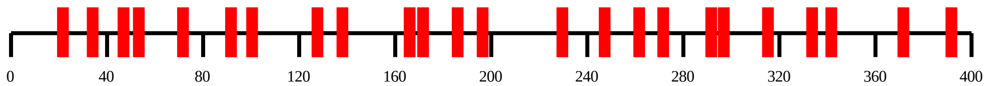
$$f(x_i, T_i) = \left(\frac{1}{T_i}\right)\sum_{i=1}^{i} x_i$$

where,

x_i = Failure count in interval "*i*"

T_i = Time at which "x_i" failures have been observed

Reliability parameters can take many forms. Table A.3-2 contrasts the differences between series and parallel reliability. Table A.3-3 provides a summary of the differences between inherent reliability and operational reliability measures.



TIME-BASED FAILURE DATA		
Failure No.	**Failure Time**	**Failure Interval**
1	19	19
2	33	14
3	44	11
4	53	9
5	75	22
6	90	15
7	98	8
8	125	27
9	135	10
10	165	30
11	170	5
12	185	15
13	196	11
14	230	34
15	245	15
16	260	15
17	270	10
18	290	20
19	294	4
20	315	21
21	333	18
22	340	7
23	370	30
24	390	20

λ = 24/390 = 0.0615 f/unit time
MTBF = 1/0.0615 = 16.26 time units

FAILURE-BASED FAILURE DATA		
Time*	**Failures in Interval**	**Cumulative Failures**
40	2	2
80	3	5
120	2	7
160	2	9
200	4	13
240	1	14
280	3	17
320	3	20
360	2	22
400	2	24

* including value as upper end-point

Convert From…	**Convert to…** Failure-Based	Time-Based
Failure-Based		• Randomly or uniformly allocate failures within the specified time intervals • Calculate time periods between adjacent failures
Time-Based	• Transform time-between-failures data to cumulative failure times • Count the number of failures whose cumulative times occur within a specified time period	

Figure A.3-1: Time-Based vs. Failure-Based Failure Data

Table A.3-2: Series and Parallel Reliability Characteristics

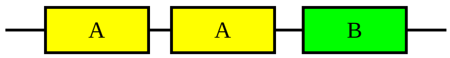
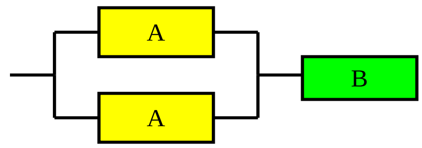
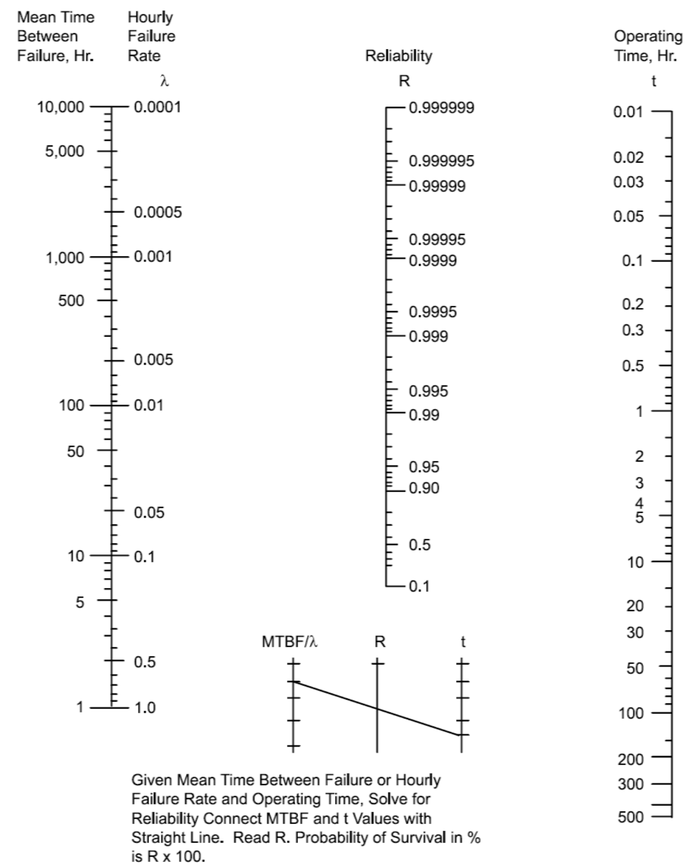
Series Reliability	**Parallel Reliability**
	
• Measure of a system's ability to operated without repair • Measured by MTBF • Recognizes effects of all occurrences that demand repair without regard to effect on overall task completion • Degraded by redundancy • Usually equal to or lower than parallel reliability because of unreliability of redundant elements	• Measure of a product's ability to complete a critical task, or set of tasks • Measured by MTBCF • Considers only failures that cause overall product failure • Improved by redundancy • Usually better than series reliability because it accounts for redundancy and other fault tolerant features

Table A.3-3: Inherent and Operational Reliability Characteristics

Inherent Reliability	**Operational Reliability**
• Used to define, measure and evaluate a design program • Derived from customer needs • Selected such that achieving it allows projected satisfaction of customer-required reliability • Expressed in inherent values such as mean-time-between-failure (MTBF) • Accounts only for failure events subject to design and manufacturing control • Includes only design and manufacturing characteristics • **Typical Terms:** – MTBF (mean-time-between-failure) – MTBCF (mean-time-between-critical-failure)	• Used to describe reliability performance when operated in expected environment • Typically not used for contractual reliability requirements (includes factors beyond the supplier's control) • Expressed in operational terms such as mean-time-between-maintenance (MTBM) • Includes combined effects of item design, quality, installation environment, preventive maintenance policy, repair, etc. • **Typical Terms:** – MTBM (mean-time-between-maintenance) – MTBR (mean-time-between-removal) – MTBCF (mean-time-between-critical-failure)

Figure A.3-2 provides a reliability nomograph based on the exponential distribution.



Reliability nomograph for the exponential failure distribution. (NAVAIR 01-1A-32, Reliability Engineering Handbook, Naval Air Systems Command, U.S. Navy, Washington, DC, 1977.)

Figure A.3-2: Reliability Nomograph for the Exponential Distribution

For More Information:

1. RADC-TR-84-25, "Reliability/Maintainability Operational Parameter Translation," Rome Laboratory, 1984

2. Fenton, N.E. and Pfleeger, S.L., “Software Metrics: A Rigorous and Practical Approach”, International Thomson Publishing, May 1998, ISBN 0534954251

3. Lyu, M.R. (Editor), “Handbook of Software Reliability Engineering”, McGraw-Hill, April 1996, ISBN 0070394008

4. Musa, J.D., “Software Reliability Engineering: More Reliable Software, Faster Development and Testing”, McGraw-Hill, July 1998, ISBN 0079132715

5. Pressman, R.S., “Software Engineering: A Practitioner’s Approach”, 5th Edition, McGraw-Hill, 1 June 2000, ISBN 0073655783

6. Schneidewind, N., “Updated Software Reliability Metrics”, Reliability Review, Vol. 29, No. 4, December 2009, ISSN 0277-9633

Appendix A.4: Software Quality Metrics

It has often been pointed out that quality means different things to different people (or even different organizations). In general, quality can be defined as the degree of excellence that can be measured in a product or system. This degree of excellence, as defined by the IEEE Std 1633 (Reference 1), can be applied to:

- the totality of features and characteristics of a software product that bear on its ability to satisfy given needs, such as conforming to specifications.
- the degree to which software possesses a desired combination of attributes.
- the degree to which a customer or user perceives that software meets his or her composite expectations.
- the composite characteristics of software that determine the degree to which the software in use will meet the expectations of the customer.

Since quality attributes can vary, it is important that, regardless of the attributes used, they must be measurable and they must meet specified user requirements. That being said, it is critical to realize that ***although a software program or system may possess good quality, it doesn't necessarily possess good reliability***. How can this happen? Generally, quality measures the success of a program against stated requirements after the software design has been completed (i.e., how well did the program do against what it was supposed to do). Good reliability practices impact a design as it is being developed (i.e., high reliability is designed in before a measurement is made as to how successful the design effort was). ***High reliability cannot effectively or efficiently be inspected or tested into a product…it must be designed in***. These axioms are apparent in a sample of possible scenarios included in Table A.4-1.

Table A.4-1: Reliability vs. Quality

Possible Scenario	Potential Impact
A required level of reliability is not specified	• Quality may be excellent (product meets all stated requirements) • Reliability may be poor (little or no emphasis on designing, inspecting or testing reliability into the product)
A required level of reliability is to be demonstrated by testing	• Quality may be excellent (product will ultimately meet reliability requirements) • Reliability may meet requirements initially (reliability not designed in, it must be grown to meet the requirement through "expensive" inspection and testing; cycles of test/fix/test to meet requirements may introduce latent faults that are not apparent until after products are shipped)
A required level of reliability is specified	• Quality may be excellent (product meets all stated requirements) • Reliability may degrade over time (maintenance meets mean time to repair requirements, but sub-optimal maintenance and repair processes may introduce latent faults)
Organization "best practices" continue to meet customer reliability needs and requirements	• Quality may be excellent (product meets customer expectations) • Reliability not used to competitive advantage (designing in higher reliability can discriminate organization from competitors to increase market share)

A candidate list of potential quality metrics was included as part of the USAF's Rome Laboratory "1994 Framework Guidebook". This list is included here as Table A.4-2, with some modification. All of the listed metrics can have either a direct or indirect impact on the level of achievable reliability for a software product or system.

Table A.4-2: USAF Rome Laboratory Software Quality Factors (Slightly Amended)

Software Quality Factor	Definition	Potential Metrics
Availability	The extent to which a system is available when needed	$\frac{\text{Total System Uptime (includes Ready Time)}}{\text{Total System Uptime + Total System Downtime}}$
Correctness	Extent to which the software conforms to specifications and standards	$\frac{\text{Total Requirements Met}}{\text{Total Requirements Specified}}$ $\frac{\text{Defects Due to Specs \& Standards}}{\text{SLOC}}$
Efficiency	Relative extent to which a resource is utilized (e.g., storage, space, processing time, communication time, etc.)	$\frac{\text{Actual Resource Utilization}}{\text{Allocated Resource Utilization}}$
Expandability	Relative effort to increase software capability or performance by enhancing current functions, or by adding new functions or data	$\frac{\text{Effort to Expand (\$\$ or person hrs)}}{\text{Effort to Develop (\$\$ or person hrs)}}$
Flexibility	Ease of effort for changing software missions, functions, or data to satisfy other requirements	(0.05)[Ave Labor Days to Change]
Integrity	Extent to which the software will perform without failure due to unauthorized access to the code or data	$\frac{\text{Defects Due to Unauthorized Access}}{\text{SLOC}}$
Interoperability	Relative effort to couple the software of one system to the software of another	$\frac{\text{Effort to Couple (\$\$ or person hrs)}}{\text{Effort to Develop (\$\$ or person hrs)}}$
Maintainability	Ease of effort for locating and fixing a software failure within a specified time period	(0.1)[Ave. Labor Days to Fix] $\frac{\text{Number of inherent failed items requiring repair}}{\text{Total time to fix all inherent failures requiring repair}}$
Portability	Relative effort to transport the software for use in another environment (hardware configuration and/or software system environment)	$\frac{\text{Effort to Transport (\$\$ or person hrs)}}{\text{Effort to Develop (\$\$ or person hrs)}}$
Reliability	Extent to which the software will perform without any failures within a specified time period	$\frac{\text{Defects Due to Inherent Failure}}{\text{SLOC}}$ $\frac{\text{Number of inherent failures experienced}}{\text{Total time period over which inherent failures were experienced}}$ $\mathbf{R} = \mathbf{e}^{-\lambda t}$ or $\mathbf{R} = \mathbf{e}^{-t/\mathrm{MTBF}}$
Reusability	Relative effort to convert a software component for use in another application	$\frac{\text{Effort to Convert (\$\$ or person hrs)}}{\text{Effort to Develop (\$\$ or person hrs)}}$
Survivability	Extent to which the software will perform/support critical functions without failure within a specified time period when a portion of the system is inoperable	$\frac{\text{Defects Due to Critical Failure}}{\text{SLOC}}$ $\frac{\text{Total time period over which inherent failures were experienced}}{\text{Total number of critical failures experienced}}$
Usability	Relative effort for using software (training and operation	$\frac{\text{Effort to Use (\$\$ or person hrs)}}{\text{Effort to Develop (\$\$ or person hrs)}}$
Verifiability	Relative ability to verify the specified software operation and performance	$\frac{\text{Effort to Verify (\$\$ or person hrs)}}{\text{Effort to Develop (\$\$ or person hrs)}}$

Figure A.4-1, taken from Reference 4, relates causes of defects and their origin for four software projects. The reader should recognize that the distribution of these defect causes is very much dependent on how an organization, and even individual projects within an organization, classify, identify and analyze their defect metrics. Of more importance than the distribution provided in this figure is the process by which defect data is captured and leveraged for improvement in quality metrics:

- Define a set of categories into which all errors/defects will be placed
- Categorize all errors/defects by origin (i.e., logic-related, standards-related, etc.)
- Record the cost associated with each error and defect
- Count and rank (in descending order) the number of errors/defects in each category
- Compute the overall cost of errors/defects in each category
- Analyze the results to identify those error/defect categories that have the highest cost impact on the organization
- Develop, implement, and verify the effectiveness of corrective action plans that will eliminate or minimize the most costly class, or classes, of errors/defects

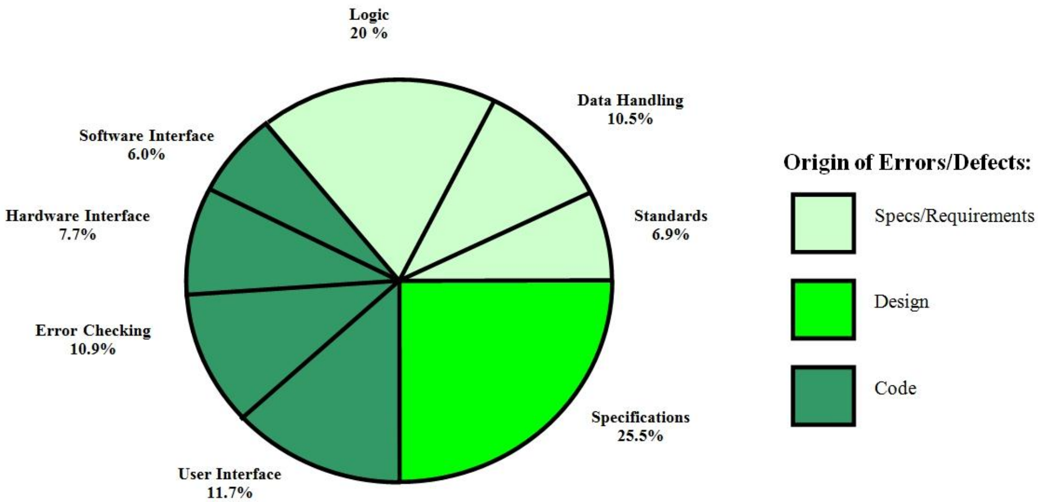


Figure A.4-1: Causes/Origins of Defects for Four Software Projects

For More Information:

1. IEEE STD 1633-2008. IEEE Recommended Practice on Software Reliability.
2. Fenton, N.E. and Pfleeger, S.L., "Software Metrics: A Rigorous and Practical Approach", International Thomson Publishing, May 1998, ISBN 0534954251
3. Grady, R.B., "Practical Software Metrics for Project Management and Process Improvement", Prentice-Hall, 1992, ISBN 0137203845
4. Grady, R.B., "Successfully Applying Software Metrics", Computer, Vol. 27, No. 9, September 1994, pp. 18-25
5. Pressman, R.S., "Software Engineering: A Practitioner's Approach", 5th Edition, McGraw-Hill, 1 June 2000, ISBN 0073655783

Appendix A.5: Relevant Statistical Concepts

Reliability Engineering is a discipline that is heavily dependent on mathematical probabilities and statistics to measure and analyze data and draw inferences about present and future performance. Appendix A.5 and its various subsections are intended to provide the reader with a very basic understanding of the statistical concepts most applicable to system and software reliability engineering. Emphasis on development of mathematical theory has intentionally been minimized. The reader is encouraged to review any of the references included in the "For More Information" section, general probability and statistics textbooks available on the market, or technical papers published in the literature if more detailed discussions on mathematical theory are desired.

Statistical techniques are a powerful, necessary and beneficial tool for analyzing data to aid in the decision-making process. That being said, their applicability in solving reliability engineering problems should not be unduly overemphasized, as they are only tools to evaluate, measure and predict reliability, i.e., **<u>the use of statistical techniques will not directly or automatically result in the initial design/development of more reliable software or systems</u>**. Table A.5-1 includes some factors to be considered in using statistical techniques.

Table A.5-1: Considerations for Applying Statistical Techniques

Consideration	Rationale
• Use the most simple statistical techniques that match the complexity of the data being collected/analyzed	• Elegant, sophisticated statistical solutions are not necessarily needed to gain a basic understanding of what the data is telling you • Start with appropriately simple techniques that match the level of detail in the data to see if reasonable interpretations of the data can be made • The use of elegant statistical techniques that are not adequately supported by sufficient data detail or quantity can result in an erroneous or confusing interpretation of results
• Use statistical techniques that match the ability of the technical and managerial staff to use, understand and interpret data analysis results	• Employing overly-sophisticated statistical techniques that exceed the technical skill of the staff can result in frustration and inconsistent application • Employing overly-sophisticated statistical techniques that provide results that are not easily explained to, or understood by, management can result in lack of management support for the techniques and misunderstanding of what the results mean
• Statistical techniques are used to assess what the system is doing currently, or what it may do in the future	• In this context, the use of statistical techniques is a reactive, rather than a proactive, approach to developing reliable systems, i.e., the data will reflect how bad/good your system development processes are based on the number of defects or problems designed in, the amount of testing that needs to be done to detect and remove them, and how many will remain when the system is delivered to the customer • Statistical techniques will not specifically indicate how to design or improve processes to reduce the number of, or eliminate, inherent software or system defects or design problems
• Statistical techniques should not be considered to be a substitute for good system reliability design processes	• Proactive software and system design reliability practices can have a more effective impact on the long-term cost-effectiveness and reliability of the system

The general areas covered within this topic area include:

- Distributions (Section A.5.1)
- Statistical Hypothesis Testing (Section A.5.2)
- Parameter Estimation (Section A.5.3)
- Confidence Bounds (Section A.5.4)

Before proceeding, however, some basic mathematic concepts should be defined and understood.

Probability: The relative frequency with which an expected output (i.e., event) will occur.

Example: The operating system of your personal computer will successfully power-down the computer, without locking up, 95 times out of 100. This can be expressed in decimal form (a probability of 0.95) or percent form (a probability of 95%).

Random Variable: A function that assigns a number to each element of a sample space, where the sample space represents the set of all possible outcomes of a random experiment. The value of that random variable is referred to as the realization of that random variable. Random variables are typically defined using upper-case letters, e.g., *X*.

Statistic: A statistic is a function of one or more random variables that do not depend on any unknown parameter. A realization of a statistic is used to summarize data (e.g., the average number of man-hours to develop 1000 lines of source code) or provides the means for making useful inferences (e.g., the number of defects remaining in a software program following testing). Examples of commonly-used statistics are $\bar{\mathbf{x}}$ .and s^2.

Stochastic Process: A process in which observations are made over a period of time, and are influenced by changes or random effects throughout the entire interval. A stochastic process is classified by the range of all of its possible values (i.e., its state space), by its index set (e.g., the index "t" can be a discrete time unit), and by the dependence among the random variables that make up the entire process.

Independent Events: The occurrence of one event has no effect on another event, i.e., the probability of another event will not increase or decrease based on the fact that the first event has occurred.

Example: The probability of successfully saving a document in a word processor program is 0.995 (probability "a"). The probability of successfully powering down the computer without having it lock up is 0.95 (probability "b"). The probability of successfully saving the document and successfully powering down the computer, given that they are independent events, is the product of the two probabilities:

$$\mathbf{P}(\text{"a" AND "b"}) = \text{Probability of both "a" and "b" happening}$$
$$\mathbf{P}(\text{"a" AND "b"}) = \mathbf{P}(\mathbf{a}) * \mathbf{P}(\mathbf{b})$$
$$\mathbf{P}(\text{"a" AND "b"}) = (0.995) * (0.95) = 0.94525$$

Mutually Exclusive Events: The occurrence of one event precludes the occurrence of another event, i.e., if the first event happens, the second event cannot happen.

Example: The probability of successfully saving a document in a word processor program is 0.995 (probability "a"). Therefore, the probability of not being able to save it is 1-0.995 = 0.005 (probability (1-"a")).

The probability of successfully powering down the computer is 0.95 (probability "b"). Therefore, the probability of the computer locking up during its power down cycle is 1-0.95 = 0.05 (probability (1-"b")).

Assuming mutually exclusive, independent events, the probability of successfully saving the document and powering down the computer:

$$\mathbf{P}(\text{"a" OR "b"}) = \text{Probability of either "a" or "b" happening, but not both}$$
$$\mathbf{P}(\text{"a" OR "b"}) = \mathbf{P}(\mathbf{a}) + \mathbf{P}(\mathbf{b}) - [(\mathbf{P}(\mathbf{a}) * \mathbf{P}(\mathbf{b})]$$
$$\mathbf{P}(\text{"a" OR "b"}) = (0.995) + (0.95) - [(0.995) * (0.95)] = 0.99975$$

Note that this success probability can also be calculated as the sum of the probabilities of both "a" and "b" being successful, plus "a" being successful but "b" failing, plus "a" failing but "b" being successful. Mathematically, this is stated as:

$$\mathbf{P}(\text{"a" OR "b"}) = [\mathbf{P}(\mathbf{a}) * \mathbf{P}(\mathbf{b})] + [\mathbf{P}(\mathbf{a}) * \mathbf{P}(1-\mathbf{b})] + [\mathbf{P}(1-\mathbf{a}) * \mathbf{P}(\mathbf{b})]$$
$$\mathbf{P}(\text{"a" OR "b"}) = [(0.995) * (0.95)] + [(0.995) * (1-.95)] + [(1-0.995) * (0.95)]$$
$$\mathbf{P}(\text{"a" OR "b"}) = (0.94525) + [(0.995) * (0.05)] + [(0.005) * (0.95)]$$
$$\mathbf{P}(\text{"a" OR "b"}) = (0.94525) + (0.04975) + (0.00475) = 0.99975$$

Dependent Events: The occurrence of one event has an effect on another event, i.e., the probability that event "b" will occur is affected by the fact that event "a" has occurred. This is defined as conditional probability.

Example: Suppose that the probability that the computer will successfully power down is partially dependent on whether a word processor document is successfully saved, i.e., suppose that 15% of the time that a document is not successfully saved the computer does not successfully power down. This conditional probability for "b" is:

$$\mathbf{P}(\mathbf{b}|\mathbf{a}) = (0.15 * 1) + [(1-0.15) * (0.95)]$$
$$\mathbf{P}(\mathbf{b}|\mathbf{a}) = 0.15 + [(0.85) * (0.95)]$$
$$\mathbf{P}(\mathbf{b}|\mathbf{a}) = 0.15 + 0.8075 = 0.9575$$

Under these conditions, the probability that a word document will successfully be saved and the computer will successfully power down is given as:

$$\mathbf{P}(\text{"a" AND "b"}) = \mathbf{P}(\mathbf{a}) * \mathbf{P}(\mathbf{b}|\mathbf{a})$$
$$\mathbf{P}(\text{"a" AND "b"}) = (0.995) * (0.9575) = 0.9527$$

Extending this to the situation where one event can have several different results, each affecting another event differently. The general equation for conditional probability, defined as Bayes' Theorem, then becomes:

$$\mathbf{P}(\mathbf{a}_1|\mathbf{b}) = \frac{\mathbf{P}(\mathbf{b}|\mathbf{a}_1) * \mathbf{P}(\mathbf{a}_1)}{\Sigma(\mathbf{P}(\mathbf{b}|\mathbf{a}_\mathbf{i}) * \mathbf{P}(\mathbf{a}_\mathbf{i}))}$$

Homogeneous Process: A process which has the property that if each variable is replaced by a constant times that variable, then the constant can be factored out.

Example: The mean value function of the Poisson process, expressed as a function of time, is $\mu(t)$. If this function is linear over time (that is, if $\mu(t) = \alpha t$ for some constant $\alpha > 0$), then the process is considered to be homogeneous.

Nonhomogeneous Process: A random process whose probability distribution varies with time. This type of process is a common assumption for many software reliability failure intensity and growth models.

Example: The mean value function of the Poisson process, expressed as a function of time, is $\mu(t)$. If this function is nonlinear over time (that is, if $\mu(t) = F_a(t)$), then the process is considered to be nonhomogeneous.

For More Information:

1. Coppola, A., "Practical Statistical Tools for the Reliability Engineer", Reliability Information Analysis Center, September 1999
2. Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008
3. Musa, J.D., "Software Reliability Engineering: More Reliable Software, Faster Development and Testing", McGraw-Hill, July 1998, ISBN 0079132715
4. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587

Appendix A.5.1: Probability Distributions

Reliability modeling draws on the mathematical theory of probability and statistics. A probability distribution represents a mathematical model that relates the quantified value of a random variable with the probability of occurrence of that value in the population from which the measurement has been drawn. Table A.5.1-1 shows specific probability distributions applicable in some situations of interest in reliability modeling.

Table A.5.1-1: Probability Distributions Applicable to Reliability Engineering

Probability Distribution	Type	Primary Uses
Binomial	Discrete	Used to find the probability of "x" events occurring in a total of "n" trials, e.g., the number of failures in a sequence of a specified number of equal-length time intervals
Poisson	Discrete	Used to model the probability of a specified number of events occurring in a specified time interval. A Poisson process can be either homogeneous or nonhomogeneous.
Exponential	Continuous	Used to describe the distribution of the time to failure when the failure rate is constant
Gamma	Continuous	Used to determine the distribution of the time by which a specified number of failures will occur when the failure rate is constant
Normal	Continuous	Used to describe the statistical mean of a sample taken from any population with a finite mean and variance
Standard Normal	Continuous	The Standard Normal distribution (Z) is derived from the Normal for ease of analysis and interpretation (mean = 0; standard deviation = 1)
Lognormal	Continuous	Used to model the time to repair and other variables in which the left tail of the distribution is truncated at some fixed finite value
Weibull	Continuous	Used to describe the distribution of failures representing constant (i.e., exponential), increasing, or decreasing failure rates, depending on the value of the slope parameter (β). Applicable only when no repair is performed following failure.
Rayleigh	Continuous	This distribution**,** among the family of Weibull distributions, is used to model the reliability of software. It addresses the expected value of defect density at different stages of the software life cycle.
Student t	Continuous	Used to test for statistical significance of the difference between the means of two samples
F Distribution	Continuous	Used to test for statistical significance of differences between the variances of two samples
Chi-Square	Continuous	A special case of the Gamma distribution, used to estimate confidence intervals around reliability test data, and to test to see whether measured data reflects a constant failure rate.

Note: These distributions apply to a version of a software system operating in an environment with an unchanging user profile.

There are two basic types of probability distributions:

Discrete distribution: When the value of a measured parameter is limited to integer values (i.e., 0, 1, 2, 3,…), the probability distribution is defined as a discrete distribution.

Example: The distribution of the number of defects remaining in software programs after 6 months of development would be a discrete distribution, since a partial defect cannot exist. Figure A.5.1-1 illustrates a discrete probability distribution.

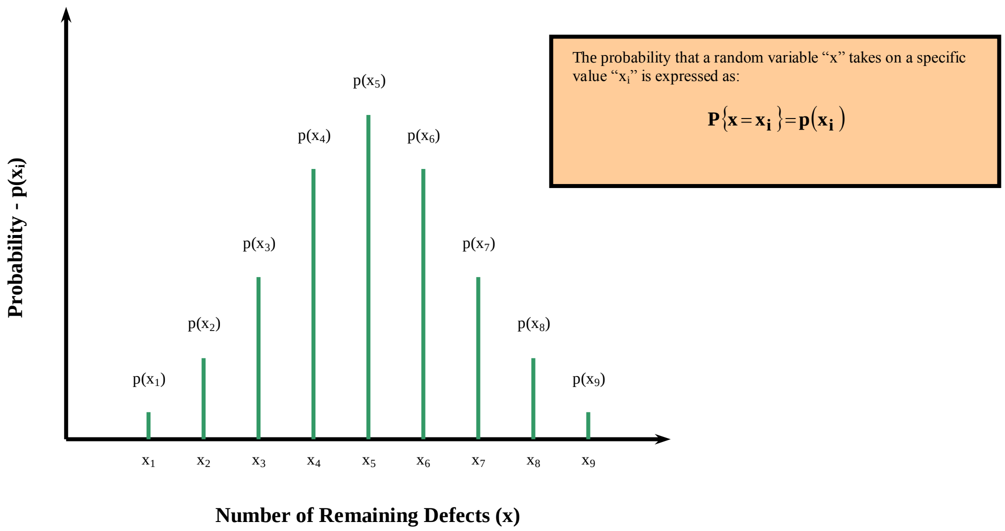


Figure A.5.1-1: Discrete Probability Distribution

Continuous distribution: When the value of a measured parameter can be expressed on a continuous scale, its probability distribution is defined as a continuous distribution.

Example: The distribution of the time to next failure would be a continuous distribution, since an infinite number of positive time values can be represented in the distribution

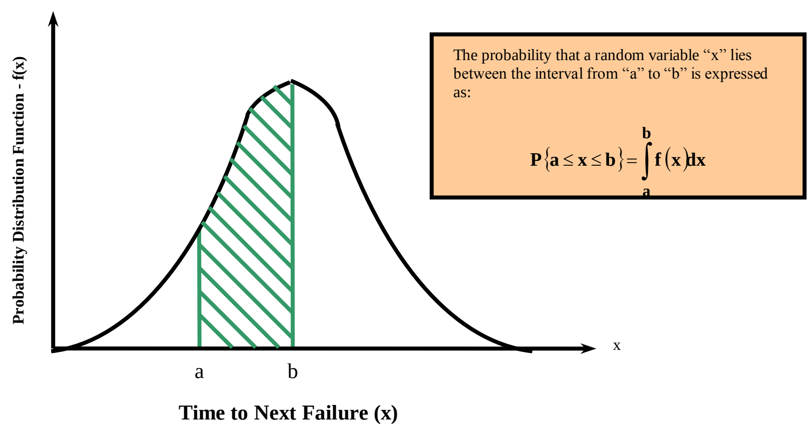


Figure A.5.1-2: Continuous Probability Distribution

A probability distribution is characterized by a probability density function (pdf). For a discrete random variable, the pdf at a given value of the random variable is the probability that the realization of the random variable will take on that value. For a continuous random variable, the area under the pdf for a given interval is the probability that a realization of the random variable will fall within that interval (Figure A.5.1-3). The probability density functions are non-negative for all values, and the sum of the probabilities over all values for discrete random variables, or the total area under the pdf for continuous random variables, always equals 1.0.

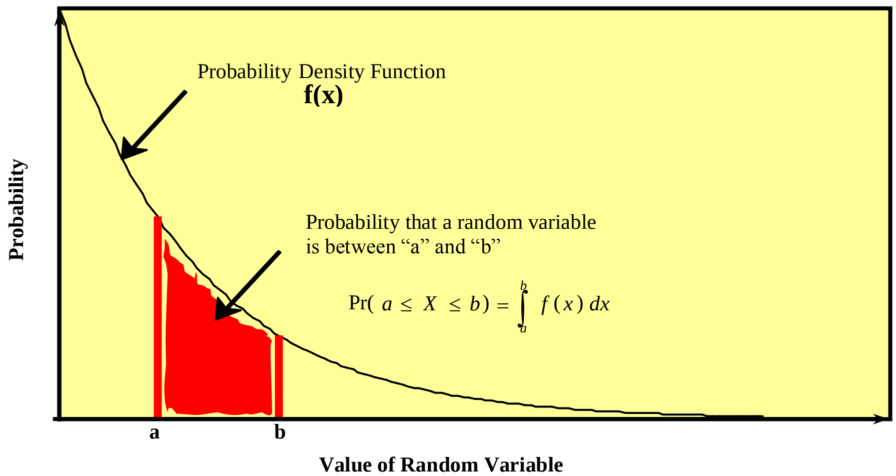


Figure A.5.1-3: A Probability Density Function (pdf)

The Cumulative Distribution Function (CDF) is the probability that the value of a corresponding random variable will not be exceeded. Figure A.5.1-4 illustrates how the CDF is determined from the pdf. Cumulative distribution functions are non-negative and non-decreasing. Given a random variable that cannot be negative, the value of the CDF at the origin is zero. The upper limit of a CDF is always 1.0, as illustrated in Figure A.5.1-5.

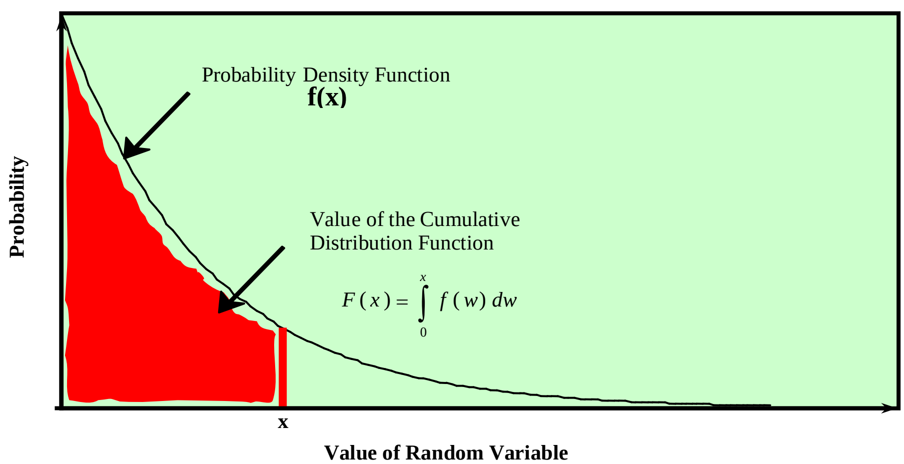
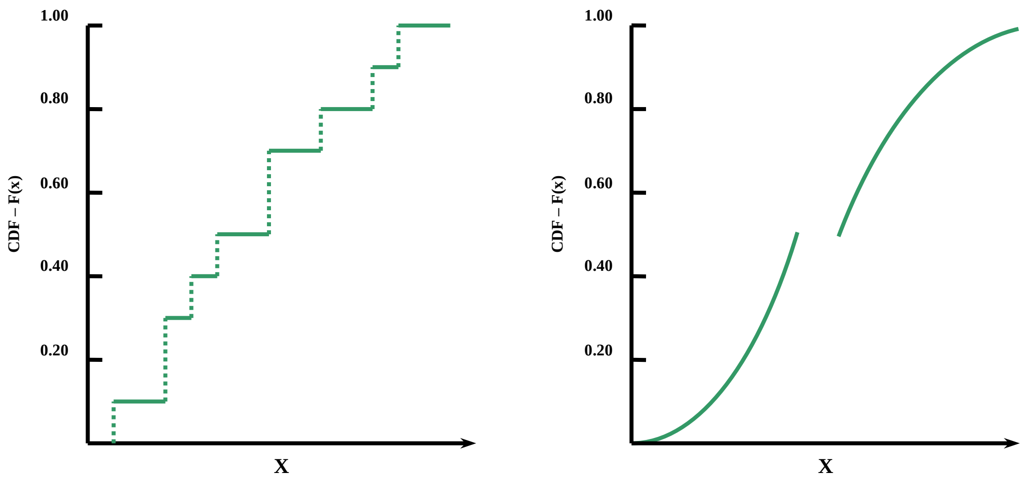


Figure A.5.1-4: Obtaining the Cumulative Distribution Function from the pdf



(a) CDF for Discrete Distributions

(b) CDF for Continuous Distributions

Figure A.5.1-5: The Cumulative Distribution Function (CDF)

Two other functions are often used to describe a random variable that represents the Time To Failure (TTF) of a system or component. The **hazard function** (also called the instantaneous failure rate) at time "*t*" is the probability that a failure will occur in a small time interval starting at time "t_i", given that no failures have occurred up to that time. The PDF and CDF can be mathematically constructed from the hazard function. The **reliability** of a system at time "*t*" is the probability that the system will operate until that time without failure. Since the CDF at time "*t*" is the probability that a failure will occur before time "*t*", the reliability function is calculated as 1.0-F(x) at the point of interest.

Most distributions used in reliability are characterized by a small number of parameters, i.e., 2 or 3. These parameters can be expressed as functions of a small number of moments of the distribution. The two most common parameters are the mean and the standard deviation of the distribution.

The method of moments is used to find parameter estimators that cannot normally be found in closed form, such as is the case with the Gamma function. In these cases, the method of moments is appropriate if an analytical relationship can be found between the moments of the variable and the parameters to be estimated.

Table A.5.1-2 provides an overview of the basic notation and mathematical representations that are common among the various types of probability distributions. The individual subsections of A.5.1 provide more detailed discussion of some of the more popular and commonly used probability distributions for software reliability.

Table A.5.1-2: Probability Distribution Notation & Mathematical Representations

Notation	Definition	Mathematical Representation
X	Random Variable	
x	Realization of a Random Variable	
$\Pr(X \in S)$	Probability That the Random Variable " X " is in the Set " S "	
$f(x)$	Probability Density Function (PDF)	$\Pr(X \in S) = \begin{cases} \sum_{x \in S} f(x), & \text{Discrete Distribution} \\ \int_S f(x)\,dx, & \text{Continuous Distribution} \end{cases}$
$F(x)$	Cumulative Distribution Function (CDF)	$F(x) = \begin{cases} \sum_{w=0}^{x} f(w), & \text{Discrete Distribution} \\ \int_0^x f(w)\,dw, & \text{Cumulative Distribution} \end{cases}$
$h(x)$	Hazard Rate	$h(x) = \frac{f(x)}{1-F(x)} = \frac{f(x)}{R(x)} = \frac{1}{R(x)}\frac{dF(x)}{dx}$
$R(x)$	Reliability	$R(x) = 1 - F(x) = \int_x^\infty f(t)\,dt = e^{-\int_0^x h(t)\,dt}$
$E[u(X)]$	Expected Value	$E[u(X)] = \begin{cases} \sum_{w=0}^{\infty} u(w)\,f(w), & \text{Discrete Distribution} \\ \int_0^\infty u(w)\,f(w)\,dw, & \text{Continuous Distribution} \end{cases}$
μ	Mean	$\mu = E(X)$
σ	Standard Deviation	$\sigma = \sqrt{E[(X-\mu)^2]}$

Note: Definitions based on the assumption that all realizations of a random variable must be non-negative.

For More Information:

1. Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008

2. Musa, J.D., "Software Reliability Engineering: More Reliable Software, Faster Development and Testing", McGraw-Hill, July 1998, ISBN 0079132715

3. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587

4. University of Alabama in Huntsville, Mathematical Sciences, http://www.math.uah.edu/stat/

Appendix A.5.1.1: Binomial Distribution

Consider a system that is operating at a number of sites with user operational profiles that are considered to be independent and identical. Assume, also, that the system operates at all sites for the same length of time. Given these assumptions, the probability that the system will operate without failure is the same across all sites. The number of sites operating without failure is represented as a random variable from a binomial distribution.

The binomial distribution arises naturally out of a number of *Bernoulli trials*. The characteristics of a Bernoulli trial are:

- Each trial result is stochastically independent from all other trial results
- Each trial can result in one of two outcomes, either success or failure
- The probability of success, *p*, is identical for each trial
- Conversely, the probability of failure is 1 – *p* (sometimes defined as "q") for each trial
- The number of successes in a total of "*n*" trials is a random variable from a binomial distribution with parameters *n* and *p*

Table A.5.1.1-1 lists the parameters for the binomial distribution probability density function (pdf), the cumulative distribution function (CDF), the mean (sometimes referred to as the expected value – E(X)), the variance, and the standard deviation.

Table A.5.1.1-1: Binomial Distribution Parameters

Parameter	Mathematical Expression
Probability Density Function (pdf)	$\mathbf{f(x)} = \binom{\mathbf{n}}{\mathbf{x}} \mathbf{p}^{\mathbf{x}} (1-\mathbf{p})^{\mathbf{n-x}}, \quad \mathbf{x} = 0, 1, 2, \ldots, \mathbf{n}$
Cumulative Distribution Function (CDF)	$\mathbf{F(x)} = \sum_{\mathbf{w}=0}^{\mathbf{x}} \binom{\mathbf{n}}{\mathbf{w}} \mathbf{p}^{\mathbf{w}} (1-\mathbf{p})^{\mathbf{n-w}}, \quad \mathbf{x} = 0, 1, 2, \ldots, \mathbf{n}$
Mean	$\mu = \mathbf{np}$
Variance	$\sigma^2 = \mathbf{np}(1-\mathbf{p})$
Standard Deviation	$\sigma = \sqrt{\mathbf{np}(1-\mathbf{p})}$

As an example, assume that an identical item of software is operating at 10 remotely located sites (each trial is stochastically independent). The site is either operating (trial failure) or down for repair (trial success). Since each site is a 24/7 operation and the <u>software is identical</u> at each site, the number of sites operating without failure is represented by a binomial distribution.

Over the last five years, 1000 trials were performed, of which 50 were "successful" (i.e., the site was found to have failed). The probability of a site having a failure over this period was calculated to be:

$$\mathbf{p} = \frac{\text{Number of "successful' trials over 5 year period}}{\text{Total number of trials over same 5 year period}}$$

$$\mathbf{p} = \frac{50}{1000} = 0.05$$

The mean number of sites that will fail over a given number of trials is:

$$\mu = \mathbf{np} = (10)(.05) = 0.50$$

The standard deviation around the mean is calculated as:

$$\boldsymbol{\sigma} = \sqrt{\mathbf{n}\,\mathbf{p}\,(1-\mathbf{p})} = \sqrt{(10)(0.05)(0.95)} = 0.6892$$

Individual binomial probabilities and cumulative binomial probabilities are typically available from tables published in a variety of mathematical and statistical textbooks.

For More Information:

1. Montgomery, D.C., "Introduction to Statistical Quality Control – 2nd Edition", John Wiley & Sons, 1991, ISBN 047151988X
2. Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987, ISBN 007044093X
3. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587

Appendix A.5.1.2: Poisson Distribution

The Poisson distribution is a widely used model for describing the number of occurrences of some event within an observed time, area, volume, code quantity, etc. General examples of how the Poisson distribution is used relate to the number of defects in the length of computer tape (obviously an old example); the number of defects in a sheet of material or length of wire; the number of failures in a repairable product over a specific time period; and the number of accesses through a network server within a certain period of time.

The Poisson distribution falls out naturally from a Homogeneous Poisson Process (HPP). The assumptions that support the idea that measured data are from an HPP are:

- The number of events occurring in non-overlapping time intervals, volumes, areas, as appropriate, are stochastically independent
- The probability of an event is the same for each interval unit of time, volume, area, etc., regardless of where that interval appears in the process
- The potential number of events is essentially unlimited (i.e., an extension of the binomial distribution where "n" is infinite)
- The probability of an event in a small interval is approximately proportional to the "length" of the interval, with proportionality constant "λ" (where "λ" is the event rate)
- The probability of two or more events in a small interval is approximately zero

Table A.5.1.2-1 lists selected random variables and their related probability distributions that are associated with a HPP process.

Table A.5.1.2-1: Distributions Associated With a Homogeneous Poisson Process

Random Variable	**Probability Distribution**
Number of Failures in Time Interval "*t*"	Poisson, with mean "λt"
Time Between Failures	Exponential, with mean θ ("$1/\lambda$")
Time to "*k*" Failures	Gamma, with shape parameter "*k*" and scale parameter "$1/\lambda$"

Table A.5.1.2-2 lists the parameters for the Poisson distribution probability density function (pdf), the cumulative distribution function (CDF), the mean (sometimes referred to as the expected value – E(X)), the variance, and the standard deviation. It should be noted that the mean and the variance of the Poisson distribution are each equal to μ (or λt, depending on the format of the model used), reflecting that the mean should be constant with time, volume, area, distance, etc.).

Table A.5.1.2-2: Poisson Distribution Parameters

Parameter	**Mathematical Expression**
Probability Density Function	$f(x) = \frac{\mu^x e^{-\mu}}{x!}, \quad x = 0, 1, 2, \ldots$
Cumulative Distribution Function	$F(x) = \sum_{x=0}^{n} \frac{\mu^x e^{-\mu}}{x!}, \quad x = 0, 1, 2, \ldots$
Mean	μ
Variance	μ
Standard Deviation	$\sigma = \sqrt{\mu}$

As an example, assume that there are, on average, 3 randomly intermittent (but very disruptive) interruptions in network service per day. What are the probabilities associated with the occurrence of service interruptions in the next 8 hours. The calculation of the mean service interruption rate is:

$$\mu = \lambda t = \frac{3\,\text{Service Interruptions}}{24\,\text{Hours}} * 8\,\text{hours}$$

$$\mu = \lambda t = (0.125) * (8) = 1.0\,\text{Interruption}$$

The standard deviation around the mean is calculated as:

$$\boldsymbol{\sigma} = \sqrt{\boldsymbol{\lambda t}} = \sqrt{1.0} = 1.0$$

Individual and cumulative Poisson probabilities are available from tables published in a variety of mathematical and statistical textbooks.

For More Information:

1. Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987, ISBN 007044093
2. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587

Appendix A.5.1.3: Normal Distribution

The normal distribution is one of the most important probability distributions in the field of statistics. In reliability, the normal distribution is most appropriately applied to the distribution of mean time between failure (MTBF) or mean time to failure (MTTF), even though actual failure rates and failure inter-arrival times for systems (not necessarily for the specific individual components that comprise them) are typically best represented by the exponential distribution.

The basic characteristics of the normal distribution are:

- The parameters of interest for the normal distribution are the mean (μ), which for MTBF or MTTF will always be $\geq$ 0, and the standard deviation (σ), which must always be positive
- The normal distribution can be applied to data from samples, even when the sampled population is not normally distributed but has a finite mean and variance, if the sample is large enough (Central Limit Theorem)
- The mean = the median = the mode (distribution symmetry)
- The binomial distribution can be approximated by the normal distribution when the number of Bernoulli trials (n) is 30 or more
- The Poisson distribution becomes approximately equal to the binomial when the number of trials (n) is high and the probability of an event (p) is low, so it can also be approximated by the normal distribution

Table A.5.1.3-1 lists the parameters for the normal distribution probability density function (pdf), the cumulative distribution function (CDF), the mean, the variance, and the standard deviation. Also included are the parameters for the standard normal distribution. Any normal probability density can be expressed in terms of the standard one as:

$$\mathbf{f}(\mathbf{x}) = \left(\frac{1}{\sigma}\right)\mathbf{Z}\left[\frac{\mathbf{x}-\mu}{\sigma}\right] \text{ and } \mathbf{F}(\mathbf{x}) = \mathbf{Z}\left[\frac{\mathbf{x}-\mu}{\sigma}\right]$$

Table A.5.1.3-1: Normal Distribution Parameters

Parameter	Mathematical Expression (Normal Distribution)	Mathematical Expression (Standard Normal Distribution)
Probability Density Function	$\mathbf{f}(\mathbf{x}) = \frac{1}{\sigma\sqrt{2\pi}}\mathbf{e}^{-\frac{(\mathbf{x}-\mu)^2}{2\sigma^2}}, \quad -\infty < \mathbf{x} < \infty$	$\mathbf{f}(\mathbf{z}) = \frac{1}{\sqrt{2\pi}}\mathbf{e}^{-\frac{(\mathbf{z})^2}{2}}, \quad -\infty < \mathbf{z} < \infty$
Cumulative Distribution Function	$\mathbf{F}(\mathbf{x}) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\mathbf{x}}\mathbf{e}^{-\frac{(\mathbf{x}-\mu)^2}{2\sigma^2}}\,\mathbf{dx}, \quad \mathbf{x} > 0$	$\mathbf{F}(\mathbf{z}) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\mathbf{z}}\mathbf{e}^{-\frac{\mathbf{z}^2}{2}}\,\mathbf{dz}, \quad -\infty < \mathbf{z} < \infty$
Mean	μ	**0**
Variance	σ^2	**1**
Standard deviation	σ	**1**
100 Pth Percentile	$\mathbf{y_p} = \mu + z_p\sigma$	
Reliability Function	$\mathbf{R}(\mathbf{x}) = 1 - \mathbf{Z}\left[\frac{\mathbf{x}-\mu}{\sigma}\right]$	

As an example, consider that a sample MTBF for a particular system has been measured to be 1000 operating hours with a known standard deviation of 250 hours. The probability that the true MTBF of the system is greater than 1200 hours is calculated as:

$$\mathbf{R}(\mathbf{x}) = \mathbf{P}\{\mathbf{X} > 1200\} = 1 - \mathbf{Z}\left[\frac{1200 - 1000}{250}\right] = 1 - \mathbf{Z}[0.80] = 1 - 0.7881 = 0.2119$$

The measured data indicate that the 10%, 50%, and 90% probabilities are:

$$\mathbf{y}_{0.10} = 1000 + (-1.28)(250) = 680 \text{ hours}$$
$$\mathbf{y}_{0.50} = 1000 + (0)(250) = 1000 \text{ hours}$$
$$\mathbf{y}_{0.90} = 1000 + (1.28)(250) = 1320 \text{ hours}$$

These results indicate that there is a 10% probability that the true MTBF of the system is $\leq$ 680 hours, a 50% probability that the true MTBF is $\leq$ 1000 hours (as you would expect), and a 90% probability that the true MTBF is $\leq$ 1320 hours.

Standard Normal probabilities are available from tables published in a variety of mathematical and statistical textbooks.

For More Information:

1. Lyu, M.R. (Editor), “Handbook of Software Reliability Engineering”, McGraw-Hill, April 1996, ISBN 0070394008
2. Madsen, R.W.; Moeschberger, M.L., “Statistical Concepts with Applications to Business and Economics”, Prentice-Hall, 1980, ISBN 0138448787
3. Montgomery, D.C., “Introduction to Statistical Quality Control – Second Edition”, John Wiley & Sons, 1991, ISBN 047151988X
4. Musa, J.D.; Iannino, A.; and Okumoto, K.; “Software Reliability: Measurement, Prediction, Application”, McGraw-Hill, May 1987, ISBN 007044093X
5. Nelson, W., “Applied Life Data Analysis”, John Wiley & Sons, 1982, ISBN 0471094587
6. Rice University – Virtual Lab in Statistics, **http://davidmlane.com/hyperstat/normal_distribution.html**

Appendix A.5.1.4: Exponential Distribution

The exponential distribution is most commonly applied in reliability to describe the times to failure for repairable items. (For non-repairable items, the Weibull distribution is popular due to its flexibility). In general, the exponential distribution has numerous applications in statistics, especially in reliability and queuing theory.

The basic characteristics of the exponential distribution are:

- It describes products whose failure rates are the same (constant) at each point in time (i.e., the "flat" portion of the reliability bathtub curve, where failures occur randomly, by "chance"). This means that if an item has survived for "t" hours, the chance of it failing during the next hour is the same as if it had just been placed in service.

- It is an appropriate distribution for software and complex systems that are comprised of different electronic and electromechanical component types, the individual failure rates of which may not follow an exponential distribution

- Since the exponential distribution is relatively easy to fit to data, it can be misapplied to data sets that would be better described using a more complex distribution

Table A.5.1.4-1 lists the parameters for the exponential distribution probability density function (pdf), the cumulative distribution function (CDF), the mean, the variance, and the standard deviation. Another useful parameter of continuous distributions is the 100 p^{th} percentile of a population, i.e., the age by which a portion of the population has failed. The 50% point is called the median and is commonly referred to as the "typical" life. The mean of the exponential distribution is roughly equal to the 63^{rd} percentile. Thus, if an item with a 1000 hour MTBF had to operate continuously for 1000 hours, there would only be a 0.37 probability of success.

As an example, consider a software system with a failure rate (λ) of 0.0025 failures per processor hour. Its corresponding mean time between failure (MTBF) is calculated as:

$$\text{MTBF} = \boldsymbol{\theta} = \frac{1}{\boldsymbol{\lambda}} = \frac{1}{0.0025} = 400 \text{ processor hours}$$

The number of processor hours by which 10%, 50%, 63.2% and 90% of the programs will have experienced a failure, respectively, is:

$$\mathbf{y}_{0.10} = -400\ln(1-0.10) = 42.14 \text{ processor hours}$$

$$\mathbf{y}_{0.50} = -400\ln(1-0.50) = 277.26 \text{ processor hours}$$

$$\mathbf{y}_{0.632} = -400\ln(1-0.632) = 399.87 \text{ processor hours}$$

$$\mathbf{y}_{0.90} = -400\ln(1-0.90) = 921.02 \text{ processor hours}$$

Table A.5.1.4-1: Exponential Distribution Parameters

Parameters	Mathematical Expression (based on failure rate)	Mathematical Expression (based on MTBF)
Probability Density Function	$f(t) = \lambda e^{-\lambda t}, \quad t > 0$	$f(t) = \frac{1}{\theta} e^{-\frac{t}{\theta}}, \quad t > 0$
Cumulative Distribution Function	$F(t) = 1 - e^{-\lambda t}, \quad t > 0$	$F(t) = 1 - e^{-\frac{t}{\theta}}, \quad t > 0$
Failure rate	λ	$\frac{1}{\theta}$
Mean	$\mu = \frac{1}{\lambda}$	$\mu = \theta$
Variance	$\sigma^2 = \frac{1}{\lambda^2}$	$\sigma^2 = \theta^2$
Standard Deviation	$\sigma = \frac{1}{\lambda}$	$\sigma = \theta$
100 P^{th} Percentile	$y_P = -\frac{1}{\lambda} \ln(1 - P)$	$y_P = -\theta \ln(1 - P)$
Reliability Function	$R(t) = e^{-\lambda t}$	$R(t) = e^{-\frac{t}{\theta}}$

The reliability function (i.e., the probability, or population fraction that survives beyond age "t") at 100 and 1000 processor hours is:

$$R(t) = e^{-(0.0025)(100)} = 0.7788 = 77.88\%$$
$$R(t) = e^{-(0.0025)(1000)} = 0.0821 = 8.21\%$$

which can be seen to be R(t) = 1 – F(t).

For More Information:

1. Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987, ISBN 007044093X
2. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587

Appendix A.5.1.5: Gamma Distribution

The gamma distribution has similar properties as those of the Weibull distribution, in that it can be made to fit or approximate a wide variety of measured data by varying its shape and scale parameters. A special case of the gamma distribution is the Chi-square distribution that, for system reliability, plays an important role in statistical testing and the construction of one- and two-sided statistical confidence limits. If the gamma shape parameter is a positive integer, the Poisson distribution models the number of occurrences of some event within a fixed time interval and the cumulative gamma distribution models the portion of that time interval required to obtain a specific number of occurrences of that same event.

It is an unfortunate circumstance in the literature, for both the gamma and the Weibull distributions, that mathematical nomenclature has not been standardized to define the important parameters of these distributions (e.g., Montgomery and Musa define the scale parameter as "λ", the failure rate, while Nelson defines it as "α", the characteristic life. The relationship in calculating the gamma mean and variance is that $\frac{1}{\lambda}=\alpha$). References 1 through 4 reflect these inconsistencies, which are summarized in Table A.5.1.5-1. Needless to say, this causes unnecessary confusion in trying to understand and communicate the characteristics of these distributions, and the reader must exercise caution when working with the mathematical expressions from various published sources. For the purposes of this Handbook, the random variable "X" will be used, with a shape parameter of "β" and a scale parameter of "α".

Table A.5.1.5-1: Confusing Terminology of the Gamma Distribution

Reference	Random Variable	Shape Parameter	Scale Parameter
Montgomery, D.C., "Introduction to Statistical Quality Control – 2nd Edition", John Wiley & Sons, 1991	X	r	λ
Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987	T	α	λ
Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982	Y	β	α
University of Alabama in Huntsville, Mathematical Sciences	X	k	b
Software-in-Systems Reliability Handbook	X	β	α

Three special cases worth noting from the gamma distribution are:

- For shape parameter = 1.0, the pdf becomes identical to the exponential distribution with the failure rate parameter "λ"
- For shape parameter = n, where "n" is an integer, the pdf becomes the *Special Erlangian* distribution which has often been used to represent service times and inter-arrival times in queuing theory. The sum of "n" exponentially distributed random variables with parameter "λ" can be expressed by this distribution
- For shape parameter = n/2 and scale parameter = ½, the pdf becomes the chi-square distribution with "n" degrees of freedom. To add to the confusion, sometimes "n" is defined in the literature as "ν", e.g., Reference 3.

The basic parameters of the gamma distribution are presented in Table A.5.1.5-2.

Table A.5.1.5-2: Gamma Distribution Parameters

Parameter	Mathematical Expression
Probability Density Function	$f(x)=\frac{1}{\Gamma(\beta)\alpha^{\beta}}x^{\beta-1}e^{-x/\alpha}, \quad x>0$
Cumulative Distribution Function	$F(x)=\frac{1}{\Gamma(\beta)\alpha^{\beta}}\int_0^x x^{\beta-1}e^{-x/\alpha}dx, \quad x>0$ $F(x)=\sum_{k=\beta}^{\infty}\frac{(x/\alpha)^k e^{-x/\alpha}}{K!}, \quad \beta>0$ and Integer
Shape parameter	β
Scale parameter	α
Mean	$\mu=\beta\alpha$
Variance	$\sigma^2=\beta\alpha^2$
Standard deviation	$\sigma=\sqrt{\beta}\,\alpha$
Reliability	$R(x)=\frac{\alpha\beta}{\Gamma(\beta)}\int_x^{\infty}x^{\beta-1}e^{-x/\alpha}dx$ (Continuous) $R(x)=\sum_{k=0}^{\beta-1}\frac{(x/\alpha)^k e^{-x/\alpha}}{k!}, \quad \beta>0$ and Integer

Figure A.5.1.5-1 provides a graphical example of the gamma distribution pdf with a variety of shape parameters. Note the exponential form of the pdf when the shape parameter is equal to 1.0.

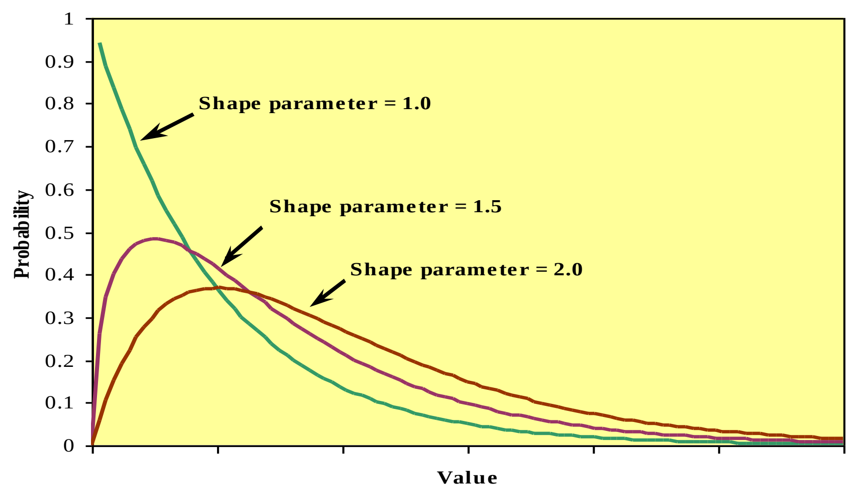


Figure A.5.1.5-1: Representative PDFs for the Gamma Distribution

As an example, consider a standby redundant system (Figure A.5.1.5-2). All three components are functionally equivalent, but not identical (i.e., if component 1 fails, component 2 or 3 will not fail). Each has an exponentially distributed characteristic life of 10,000 operating hours. While component 1 operates, the other two are bypassed. A checking algorithm (the "switch") samples the component 1 output. If it is incorrect, the algorithm uses component 2. If that output is also incorrect, it switches to component 3.

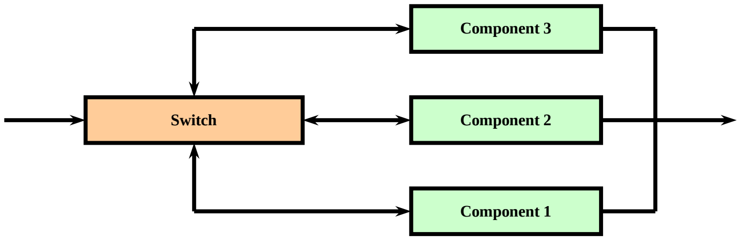


Figure A.5.1.5-2: A "Standby Redundant" System

The system life is gamma distributed with a shape parameter of 3 (the number of components in the system) and a scale parameter of 10,000 hours. The calculated system life mean, standard deviation, and reliability over a 24-hour period is:

$$\mu = \beta\alpha = (3)*(10{,}000) = 30{,}000 \text{ operating hours}$$

$$\sigma = \sqrt{\beta}\,\alpha = \sqrt{3}*10{,}000 = 17{,}321 \text{ operating hours}$$

$$\mathbf{R}(24) = \sum_{\mathbf{k}=0}^{2} \frac{[(24/10000)]^{\mathbf{k}}\,\mathbf{e}^{-(24/10000)}}{\mathbf{k}!} = 0.99760 + .00239 + .00000 = 0.99999$$

Values for the gamma function can be obtained from tables or from web-based gamma function calculators (e.g., "http://www.efunda.com/math/gamma/findgamma.cfm"). Values of the gamma function are calculated as:

$$\Gamma(\beta + \mathbf{x}) = (\beta - 1 + \mathbf{x})*(\beta - 2 + \mathbf{x})*(\ldots)*(1 + \mathbf{x})*\Gamma(1 + \mathbf{x})$$

e.g., if $\beta = 3$ and $x = 0.15$, then $\Gamma(3.15) = (2.15)*(1.15)*\Gamma(1.15)$

For More Information:

1. Montgomery, D.C., "Introduction to Statistical Quality Control – 2nd Edition", John Wiley & Sons, 1991, ISBN 047151988X
2. Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987, ISBN 007044093X
3. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587

Appendix A.5.1.6: Weibull Distribution

The Weibull distribution has become increasingly important in the reliability discipline since it represents a general distribution which, through measurement of its distribution parameter values, can model a wide range of item life characteristics. It can accommodate increasing, decreasing and constant failure rates. Weibull analysis assumes that there has been no repair of failed items and is most effective for modeling single failure modes/mechanisms, rather than mixed modes/mechanisms.

The basic features of the Weibull are:

- The shape parameter, β, which describes the shape of the PDF.
- The scale parameter, α, is a value that occurs at the 63rd percentile of the distribution and is called the characteristic life.
- The location parameter, γ, is the value that represents the failure free period for the equipment. If an item does not have a period where the probably of failure is zero, then $\gamma = 0$ and the Weibull distribution becomes a two parameter distribution.
- Determination of β, η, and γ can easily be estimated using Weibull probability paper or by using available Weibull software programs.
- The Weibull can be used to determine the points on the bathtub curve where the failure rate is changing from decreasing, to constant, to increasing.
- The Weibull can be used to determine what other distribution a set of data may follow.

There are two general classes of the Weibull distribution, the first being the two-parameter Weibull and the second being the three-parameter Weibull. The two-parameter Weibull uses a shape parameter that reflects the tendency of the failure rate (increasing, decreasing, or constant) and a scale parameter that reflects the characteristic life of items being measured (≅ 63.2% of the population will have failed). The three-dimensional Weibull adds a location parameter used to represent the minimum life of the population (e.g., a failure mode that does not immediately cause system failure at time zero, such as a software algorithm whose degrading calculation accuracy does not cause system failure until four calls to the algorithm have been made). Note that in most cases, the location parameter is set to zero (failures assumed to start at time zero) and the Weibull distribution reverts to the two-dimensional case.

As with the gamma distribution, the definition of Weibull parameters is inconsistent throughout the literature. Table A.5.1.6-1 illustrates how some sources define these parameters.

Table A.5.1.6-1: Confusing Terminology of the Weibull Distribution

Reference	Weibull Form	Random Variable	Shape Parameter	Scale Parameter	Location Parameter
Montgomery, D.C., "Introduction to Statistical Quality Control – 2nd Edition", John Wiley & Sons, 1991	**3-P**	***X***	β	δ	γ
Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987	**2-P**	***T***	α	β	
Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982	**2-P**	***Y***	β	α	
University of Alabama in Huntsville, Mathematical Sciences	**2-P**	***X***	***k***	***b***	
MIL-HDBK-338, Section 5.3.6	**3-P**	***T***	β	η	γ
Software-in-Systems Reliability Handbook	**2-P**	***X***	β	α	

Special cases worth noting from the Weibull distribution follow. For much life data, the Weibull distribution is more suitable than the exponential, normal and extreme value distributions, so it should be the distribution of first resort.

- For shape parameter < 1.0, the Weibull pdf takes the form of the gamma distribution (see Section 3.7.1.4) with a decreasing failure rate (i.e., infant mortality)
- For shape parameter = 1.0, the failure rate is constant so that the Weibull pdf takes the form of the simple exponential distribution with failure rate parameter "λ" (the flat part of the reliability bathtub)
- For shape parameter = 2.0, the Weibull pdf takes the form of the lognormal or *Rayleigh* distribution, with a failure rate that is linearly increasing with time (i.e., wear-out)
- For 3 $\leq$ shape parameter $\leq$ 4, the Weibull pdf approximately takes the form of the Normal distribution
- For shape parameter $\geq$ 10, the Weibull distribution is close to the shape of the smallest extreme value distribution (not covered in this Toolkit)

The basic parameters of the 2-parameter Weibull distribution are presented in Table A.5.1.6-2. To have the mathematical expressions reflect a 3-parameter Weibull, replace all values of "x" with "(x-x_0)".

Table A.5.1.6-2: Weibull Distribution Parameters

Parameter	Mathematical Expression
Probability Density Function	$f(x) = \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{\beta-1} e^{\left[-\left(\frac{x}{\alpha}\right)^{\beta}\right]}, \quad x > 0$
Cumulative Distribution Function	$F(x) = 1 - e^{\left[-\left(\frac{x}{\alpha}\right)^{\beta}\right]}$
Shape parameter	β
Scale parameter	α
Failure Rate	$\lambda(x) = \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{\beta-1}$
Mean	$\mu = \alpha\Gamma\left(1+\frac{1}{\beta}\right)$
Variance	$\sigma^2 = \alpha^2\left[\Gamma\left(1+\frac{2}{\beta}\right) - \Gamma\left(1+\frac{1}{\beta}\right)^2\right]$
Standard deviation	$\sigma = \alpha\left[\Gamma\left(1+\frac{2}{\beta}\right) - \Gamma\left(1+\frac{1}{\beta}\right)^2\right]^{0.5}$
100 P^{th} Percentile	$y_P = \alpha[-\ln(1-P)]^{1/\beta}$
Reliability	$R(x) = e^{\left[-\left(\frac{x}{\alpha}\right)^{\beta}\right]}$

Figure A.5.1.6-1 provides a graphical example of the Weibull distribution pdf with a variety of shape parameters. Note the exponential form of the pdf when the shape parameter is equal to 1.0 and the Normal shape of the pdf when the shape parameter is 3.5.

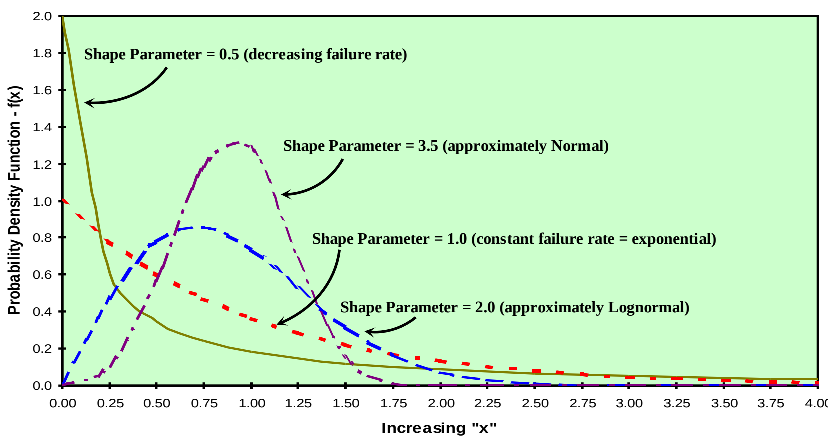


Figure A.5.1.6-1: Example PDFs for Weibull Distribution

As an example, consider that very early in the system integration phase of a large software development effort, there have been numerous failures due to software that have caused the system to crash (the predominant system failure mode). Plotting the failure times of this specific failure mode (other failure modes are ignored for now) on Weibull probability paper resulted in a shape parameter value of 0.77 and a scale parameter value of approximately 32 hours. Based on these parameters, the calculated reliability and failure rate of the software at 10 system hours is expected to be:

$$\lambda(10) = \frac{0.77}{32}\left(\frac{10}{32}\right)^{0.77-1} = 0.0314 \text{ failures per hour}$$

$$R(10) = e^{\left[-\left(\frac{10}{32}\right)^{0.77}\right]} = 0.6647$$

For More Information:

1. Montgomery, D.C., "Introduction to Statistical Quality Control – 2nd Edition", John Wiley & Sons, 1991, ISBN 047151988X
2. Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987, ISBN 007044093X
3. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587
4. Shooman, M., "Probabilistic Reliability, An Engineering Approach," McGraw-Hill, 1968
5. Abernethy, R.B., "The New Weibull Handbook", Gulf Publishing Co., 1994

Appendix A.5.1.7: Rayleigh Distribution

The Rayleigh distribution is a special case of the Weibull distribution, with a Weibull shape parameter of $\beta = 2.0$. If software errors are found to be best represented by the Rayleigh distribution, then its basic parameters are presented in Table A.5.1.7-1. Note that the failure rate will not be constant over time.

Table A.5.1.7-1: Rayleigh Distribution Parameters

Parameter	Mathematical Expression
Probability Density Function	$f(x;\sigma) = \frac{x}{\sigma^2} \exp\left(\frac{-x^2}{2\sigma^2}\right)$
Cumulative Distribution Function	$1 - \exp\left(\frac{-x^2}{2\sigma^2}\right)$
Scale parameter	σ
Relationship Between Rayleigh Scale Parameter (σ) and Weibull Scale Parameter (α)	$\sigma = \frac{\alpha}{\sqrt{2}}$
Failure Rate	$\lambda(x) = \frac{x}{\sigma^2}$
Mean	$\sigma\sqrt{\frac{\pi}{2}}$
Variance	$\frac{4-\pi}{2}\sigma^2$
Standard deviation	$\sqrt{\frac{4-\pi}{2}\sigma^2}$
100 Pth Percentile	$\sigma\sqrt{2}\left[-\ln(1-P)\right]^{0.50}$
Reliability	$R(x) = e^{-\left(\frac{x}{\sigma\sqrt{2}}\right)^2}$

Figure A.5.1.7-1 provides a graphical example of the Rayleigh distribution pdf with a variety of scale parameters. Figure A.5.1.7-2 provides a graphical example of the Rayleigh distribution CDF with a variety of scale parameters.

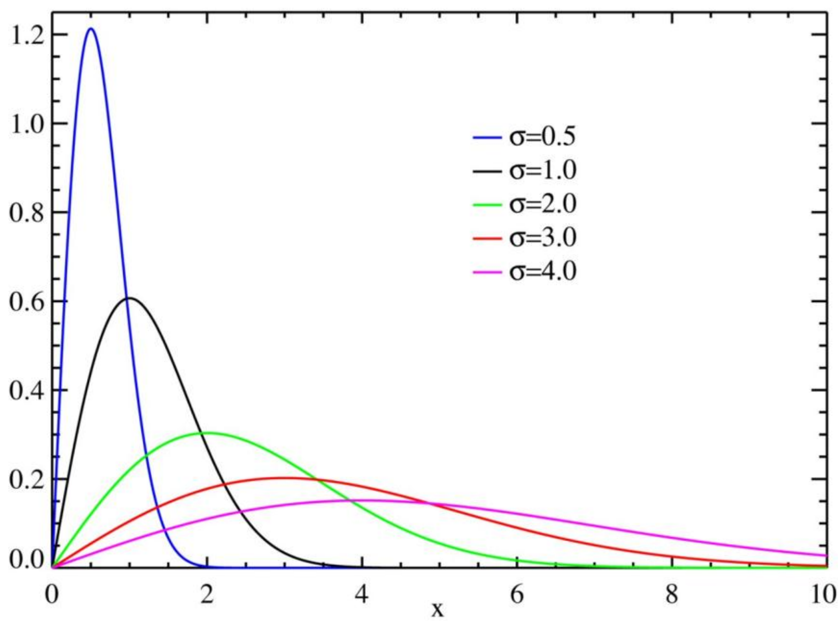


Figure A.5.1.7-1: Example PDFs for the Rayleigh Distribution

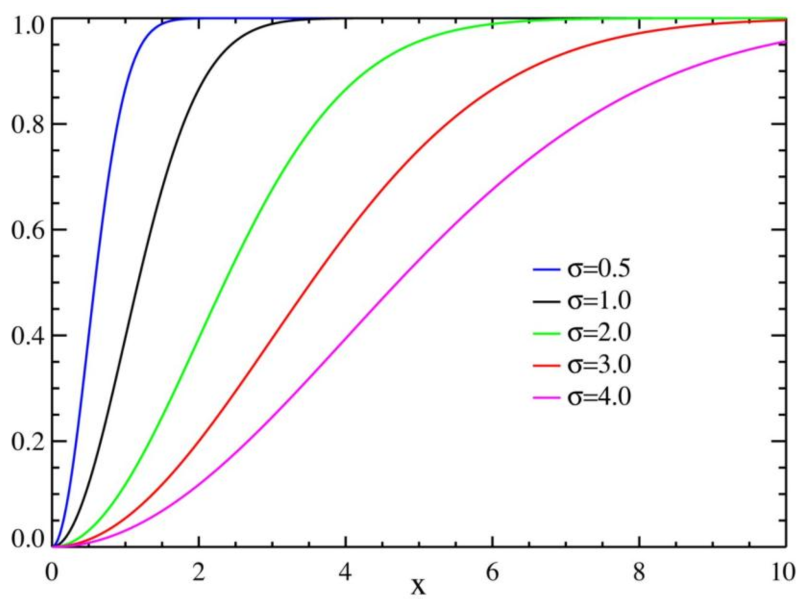


Figure A.5.1.7-2: Example CDFs for the Rayleigh Distribution

The Rayleigh distribution exhibits a linearly increasing hazard function as a function of time. The implication is when Time-to-Failure (TTF) follows the Rayleigh distribution, there is an ageing or wear-out process in effect and failures do not satisfy the requirements of a stationary random process. During the early life of a component, where a hazard rate is significant, the probability of failure-free operation will decrease as a function of time more slowly than if the hazard function was based on the exponential distribution. As time increases, the probability of failure-free operation decreases at a faster rate than with the exponential distribution. This distribution is very useful in modeling rapidly deteriorating software performance.

Two basic assumptions associated with the Rayleigh model when applied to software reliability defect rates are:

- The defect rate observed during the software development process is a reflection of the defect rate observed in the field (positive correlation)
  - The higher the Rayleigh curve, the higher the field defect rates
  - This phenomena is related to the concept of error injection
- Given the same error rejection rate, if more software defects are discovered and removed earlier, there will be fewer defects remaining at later phases of the development cycle

A basic output of the Rayleigh model for software applications, then, is the expected latent fault density in the software code at the time it is released.

The following categories are typically used to prioritize approaches when a Rayleigh analysis is being performed:

- **Critical (Priority 1):** An error that either (a) prevents the completion of an operational or mission-essential function, or (b) interferes with system performance to the extent that it prevents completion of a mission-essential function, or (c) jeopardizes personnel safety
- **Major (Priority 2):** An error that adversely impacts completion of a mission-essential function due to performance degradation for which no alternative functionality is provided. Rebooting/restarting the software is not an acceptable alternative since it is represents unacceptable interference with, or interruption of, system use.
- **Minor (Priority 3):** An error that adversely impacts completion of an operational or mission-essential function due to performance degradation for which a reasonably suitable alternative is provided. Rebooting/restarting the software is not an acceptable alternative due to its interference with, or interruption of, system use.
- **Annoyance (Priority 4):** An error which results in an inconvenience or annoyance to the operator, but has no impact on the completion of an operational or mission-essential function
- **Other (Priority 5):** All other errors not defined above

For More Information:

1. Elsayed, E.A., “Reliability Engineering”, Addison Wesley Longman, 1996, ISBN 0201634813
2. http://en.wikipedia.org/wiki/Rayleigh_distribution
3. Peterson, J.R., “Software Reliability Applications”, 2010 Annual Reliability and Maintainability Symposium, Tutorial Notes, January, 2010

Appendix A.5.2: Statistical Hypothesis Testing

Statistics involves drawing inferences from realizations of random variables, such as observed failure times. Typical inferences consist of point and interval estimates of distribution parameters and decisions based on statistical hypothesis testing.

A statistical hypothesis represents a statement about the probability distribution of a random variable, or about the value(s) of one or more distribution parameters. Statistical hypothesis testing provides a framework for decisions based on observed sample data and partial information when distribution parameters of the entire data set are not known. The basic definitions that apply to statistical hypothesis testing are contained in Table A.5.2-1.

Table A.5.2-1: Basic Terminology Used in Statistical Hypothesis Testing

Term	Definition
Null Hypothesis (H_0)	The default hypothesis, which is typically established to either (1) demonstrate that a product surpasses a requirement (or the performance of other products) or (2) assess that a product parameter is consistent with a specified value, or whether corresponding parameters of a number of products are comparable (where "parameter" means any distribution value, including percentiles and reliabilities).
Alternative Hypothesis (H_1)	The hypothesis that is to be accepted if the null hypothesis is rejected.
Type I Error	An incorrect decision in which the null hypothesis is true, but is rejected (see Producer's/Supplier's Risk).
Type II Error	An incorrect decision in which the null hypothesis is not true, but is accepted (see Consumer's Risk).
Sample Size	The number of random variables from which a statistic is calculated. Generally, the Consumer Risk is a function of sample size, i.e., as sample size increases the Consumer Risk decreases.
Significance Level	The exact probability, expressed as a percentage, of the null distribution beyond the observed statistic (i.e., erroneous rejection of the null hypothesis). If the observed statistic is beyond the upper or lower 5% point, it is statistically significant. If it is beyond the 1% point, it is highly statistically significant. If it is beyond the 0.1% point, it is very highly statistically significant.
Power (1-β)	The probability, which may be expressed as a percentage, of correctly rejecting the null hypothesis, given that the null hypothesis reflects, as an example, the correct distribution, or a good system under test. $$\mathbf{Power} = 1 - \boldsymbol{\beta} = \mathbf{P}\{\text{reject } H_0 \mid H_0 \text{ is false}\}$$
Consumer's Risk (β)	The probability, which may be expressed as a percentage, of erroneously accepting the null hypothesis when the alternative hypothesis is correct (e.g., accepting a bad system that you thought was "good"). Related to power for a test in which the null hypothesis is that the system under test is a good system. $$\boldsymbol{\beta} = \mathbf{P}\{\text{Type II Error}\} = \mathbf{P}\{\text{accept } H_0 \mid H_0 \text{ is false}\}$$
Producer's/Suppliers Risk (α)	The probability, which may be expressed as a percentage, of erroneously rejecting the null hypothesis when the null hypothesis is correct (e.g., rejecting a good system that you thought was "bad"). $$\boldsymbol{\alpha} = \mathbf{P}\{\text{Type I Error}\} = \mathbf{P}\{\text{reject } H_0 \mid H_0 \text{ is true}\}$$
Critical/Rejection Region	The set of values of a test statistic that lead to the rejection of the null hypothesis.
One-Sided Hypothesis	A hypothesis in which a parameter value from the alternative hypothesis is greater than (or less than) the corresponding parameter value from the null hypothesis
Two-Sided Hypothesis	A hypothesis in which a parameter of the null hypothesis has a specified value, or parameters of different populations are equal. The alternative hypothesis is that they are not equal to the value of the parameter from the null hypothesis.

Some examples of hypothesis tests that may be appropriate for system reliability work are provided in Table A.5.2-2, showing the null hypothesis, the alternative hypothesis, and whether the hypothesis represents a one-sided or two-sided test.

Table A.5.2-2: Examples of Hypothesis Tests

Null Hypothesis	Alternative Hypothesis	One- or Two-Sided
1. The mean of an exponential distribution exceeds a specified value	The mean of an exponential distribution is less than or equal to a specified value	One-Sided
2. Product reliability at a specified point in time exceeds a given value	Product reliability at a specified point in time is less than or equal to a given value	One-Sided
3. A Weibull shape parameter equals 1.0, i.e., product life has an exponential distribution	A Weibull shape parameter does not equal 1.0, i.e., product life does not have an exponential distribution	Two-Sided
4. The means of a number of exponential distributions are equal	The means of a number of exponential distributions are not equal	Two-Sided
5. The shape parameters of a number of Weibull distributions are equal	Two or more of the shape parameters of a number of Weibull distributions are not equal	Two-Sided
6. The specific percentiles of a number of Weibull distributions are equal	The specific percentiles of a number of Weibull distributions are not equal	Two-Sided
7. A specific model fits the observed data using a goodness-of-fit test	The specific model does not fit the observed data using a goodness-of-fit test	Two-Sided
8. A software system undergoing test is a "good" system for achieving a specific level of reliability	The software system undergoing test is not a good system for achieving the specified level of reliability	One-Sided

The framework of statistical hypothesis testing is provided in Figure A.5.2-1.

Reality / Decision	Null Hypothesis H_0	Alternative Hypothesis H_1
Accept H_0	Correct Decision Probability = 1 - α	Type II Error Probability = β
Reject H_0	Type I Error Probability = α	Correct Decision Probability = 1 - β

Reality / Decision	Null Hypothesis Failure Intensity is Constant	Alternative Hypothesis Failure Intensity is Trending
Accept H_0 as True	Correct Decision Probability = 1 - α	Type II Error Probability = β
Reject H_0 as False	Type I Error Probability = α	Correct Decision Probability = 1 - β

Reality / Decision	Null Hypothesis (Bush Won Florida)	Alternative Hypothesis (Bush Did Not Win Florida)
Accept Bush as Winner	Correct Decision Bush is President	Type II Error Gore Should Be President
Reject Bush as Winner	Type I Error Bush Should Be President	Correct Decision Gore is President

Figure A.5.2-1: Framework and Examples for Statistical Hypothesis Testing

The ramifications of the Type I and Type II errors that arise in hypothesis testing should always be assessed to determine their impact on safety, reliability, cost, etc., before the null hypothesis is defined. The probability that the test will reject the null hypothesis when the null hypothesis is in fact true is called the *significance level*. Typical significance levels used, expressed as a percent (100 α %), are 10%, 5%, and 1%. A lower percentage implies higher significance. Therefore, the rejection of the null hypothesis at a higher significance level is less likely when the null hypothesis is true.

The hypothesis that is ultimately accepted is based on a statistic, where the value of the statistic is calculated from a realization of the random variables. Given the significance level and the probability distribution of the statistic under the model specified by the null hypothesis, one can calculate a *critical/rejection region*, which is a set of values of the statistic such that the significance level is the probability of the statistic being in the critical region under the null hypothesis. In practice, one chooses the significance level based on the needs of the business, collects the data required to generate the required statistic, calculates the statistic, and rejects the null hypothesis if (and only if) the value of the statistic lies in the critical region. The process steps are illustrated in Table A.5.2-3.

Table A.5.2-3: Steps in Statistical Hypothesis Testing

Sequence of Steps	**Comments**
1. State the null hypothesis, H_0, and the alternative hypothesis, H_1	Decide whether to use a one- or two-sided test alternative. If a one-sided alternative is used, carefully consider the direction of the inequality.
2. Specify a significance level, α	Common values of α are 0.05 or 0.01, depending on the seriousness of the impact of committing a Type I error. Other values of higher or lower significance can be used.
3. Specify a sample size, n	The number of samples used may be dictated by time/cost constraints, or the number may be chosen to achieve specific error probabilities.
4. Select an appropriate test statistic	Generally, the test statistic will be standardized. For parametric tests, the test statistic will typically be the sample counterpart of the parameter being tested.
5. Define the region of rejection (critical region)	The critical region is usually bounded by the percentiles of the standardized test statistic.
6. Compute the value of the statistic and determine whether the null hypothesis should be accepted or rejected.	If the calculated value of the test statistic is in the critical region, reject H_0. Otherwise, accept H_0.

Failing to reject the null hypothesis when the alternative hypothesis is true is a Type II error. The probability that the null hypothesis will be rejected under the alternative hypothesis is known as the power of the test. In other words, the power is the probability of not committing a Type II error. Power is a function of the statistic, the significance level, and the sample size. The sample size is determined given the statistic, the alternative hypothesis, the significance level, and the desired power. Increasing the sample size increases the power of the test and, therefore, reduces the probability of a Type II error (reduces Consumer's Risk).

Depending on the type of hypothesis-testing problem encountered, there is a test statistic that can be defined to determine the critical value that serves as the basis for accepting or rejecting the null hypothesis. Figures A.5.2-2 through A.5.2-6 provide a flowchart representing how a test statistic might be chosen given a specific hypothesis testing scenario.

Does hypothesis test concern a proportion "p", success probability "p", or defect rate "λ"?

YES → **One Sample? H_0: $p = p_0$ (Binomial) H_0: $\lambda = \lambda_0$ (Poisson)**

NO → Go to Figure 3.5.2-3

One Sample? YES →

Hypothesis	Test Statistic	Rejection Criteria
Binomial (normal approximation)		
H_0: $p = p_0$ H_1: $p \neq p_0$	$Z_0 = \begin{cases} \dfrac{(x+0.5)-np_0}{\sqrt{np_0(1-p_0)}} & \text{if } x < np_0 \\ \dfrac{(x-0.5)-np_0}{\sqrt{np_0(1-p_0)}} & \text{if } x > np_0 \end{cases}$	$\lvert Z_0 \rvert > Z_{\alpha/2}$
H_0: $p = p_0$ H_1: $p < p_0$		$Z_0 < -Z_\alpha$
H_0: $p = p_0$ H_1: $p > p_0$		$Z_0 > Z_\alpha$
Binomial (one sample each from two populations) **If H_0 true, then $p_1 = p_2 = p$**		
H_0: $p_1 = p_2$ H_1: $p_1 \neq p_2$	$\hat{p} = \dfrac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2}$ $Z_0 = \dfrac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})\left(\dfrac{1}{n_1} + \dfrac{1}{n_2}\right)}}$	$\lvert Z_0 \rvert > Z_{\alpha/2}$
H_0: $p_1 = p_2$ H_1: $p_1 < p_2$		$Z_0 < -Z_\alpha$
H_0: $p_1 = p_2$ H_1: $p_1 > p_2$		$Z_0 > Z_\alpha$
Poisson (normal approximation, "n" is large)		
H_0: $\lambda = \lambda_0$ H_1: $\lambda \neq \lambda_0$	$Z_0 = \dfrac{\bar{x} - \lambda_0}{\sqrt{\lambda_0 / n}}$	$\lvert Z_0 \rvert > Z_{\alpha/2}$
H_0: $\lambda = \lambda_0$ H_1: $\lambda < \lambda_0$		$Z_0 < -Z_\alpha$
H_0: $\lambda = \lambda_0$ H_1: $\lambda > \lambda_0$		$Z_0 > Z_\alpha$

One Sample? NO → **Two or more samples?** YES →

Hypothesis	Test Statistic	Rejection Criteria
Comparison of multiple samples from one Poisson distribution		
H_0: Data are generated from a Poisson distribution with mean "μ" H_1: Data are not generated from a Poisson distribution with mean "μ"	General Form: $\chi_0^2 = \sum_{i=1}^{n} \dfrac{(O_i - E_i)^2}{E_i}$ where, O_i = observed number of occurrences E_i = expected number of occurrences based on assumed distribution For Poisson Distribution: $\chi_0^2 = \dfrac{\sum_{i-1}^{n}\left(\lambda_i - \bar{\lambda}\right)^2}{\bar{\lambda}}$ where, λ_i = number of observed defects from the i^{th} sample $\bar{\lambda}$ = average number of defects from all of the samples taken Degrees of freedom: Product (r-1)(c01) where, r = # of rows of data c = # of columns of data	$\chi_0^2 > \chi^2_{\alpha,(r-1)(c-1)}$

Figure A.5.2-2: Hypothesis Test Scenario for Discrete Distributions

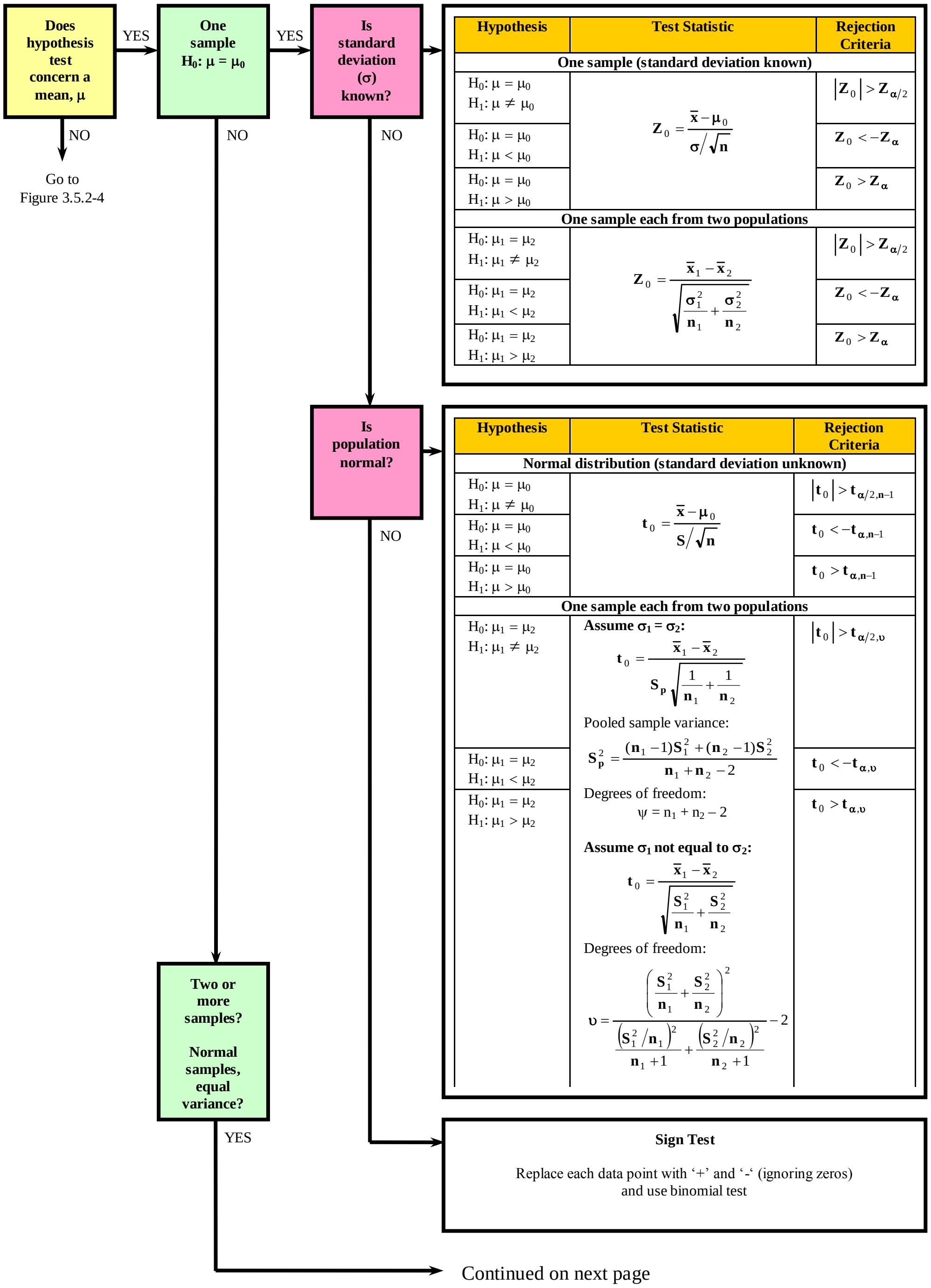
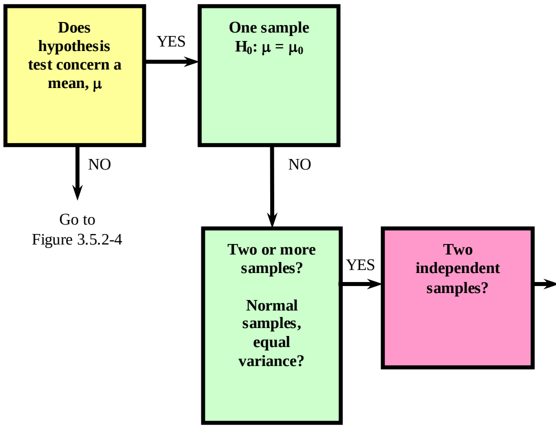
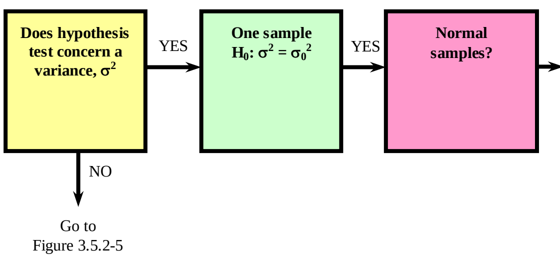


Figure A.5.2-3: Hypothesis Test Scenario for Distribution Means



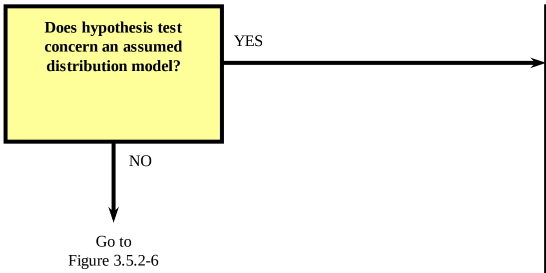
Hypothesis	Test Statistic	Rejection Criteria
Two independent samples from same population		
H_0: $\mu_X = \mu_Y$ H_1: $\mu_X \neq \mu_Y$	**Assume $\sigma_1 = \sigma_2$:** $t_0 = \dfrac{\bar{X} - \bar{Y}}{S_p\sqrt{\dfrac{1}{n_X} + \dfrac{1}{n_Y}}}$ Pooled sample variance: $S_p^2 = \dfrac{(n_X - 1)S_X^2 + (n_Y - 1)S_Y^2}{n_X + n_Y - 2}$ Degrees of freedom: $\psi = n_X + n_Y - 2$	$\lvert t_0 \rvert > t_{\alpha/2,\upsilon}$
H_0: $\mu_X = \mu_Y$ H_1: $\mu_X < \mu_Y$		$t_0 < -t_{\alpha,\upsilon}$
H_0: $\mu_X = \mu_Y$ H_1: $\mu_X > \mu_Y$		$t_0 > t_{\alpha,\upsilon}$

Figure A.5.2-3: Hypothesis Test Scenario for Distribution Means (continued)



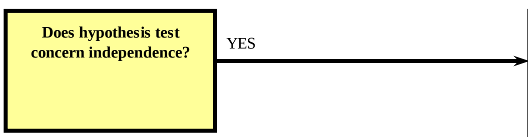
Hypothesis	Test Statistic	Rejection Criteria
Two independent samples from same population		
H_0: $\sigma^2 = \sigma_0^2$ H_1: $\sigma^2 \neq \sigma_0^2$	$\chi_0^2 = \dfrac{(n-1)S^2}{\sigma_0^2}$	$\chi_0^2 > \chi^2_{\alpha/2,n-1}$ or $\chi_0^2 < \chi^2_{1-\alpha/2,n-1}$
H_0: $\sigma^2 = \sigma_0^2$ H_1: $\sigma^2 < \sigma_0^2$		$\chi_0^2 < \chi^2_{1-\alpha,n-1}$
H_0: $\sigma^2 = \sigma_0^2$ H_1: $\sigma^2 > \sigma_0^2$		$\chi_0^2 > \chi^2_{\alpha,n-1}$
One sample each from two populations		
H_0: $\sigma_1^2 = \sigma_2^2$ H_1: $\sigma_1^2 \neq \sigma_2^2$	$F_0 = \dfrac{S_1^2}{S_2^2}$	$F_0 > F_{\alpha/2,n1-1,n2-1}$ or $F_0 < F_{1-\alpha/2,n1-1,n2-1}$
H_0: $\sigma_1^2 = \sigma_2^2$ H_1: $\sigma_1^2 < \sigma_2^2$	$F_0 = \dfrac{S_2^2}{S_1^2}$	$F_0 > F_{\alpha,n1-1,n2-1}$
H_0: $\sigma_1^2 = \sigma_2^2$ H_1: $\sigma_1^2 > \sigma_2^2$	$F_0 = \dfrac{S_1^2}{S_2^2}$	$F_0 > F_{\alpha,n1-1,n2-1}$

Figure A.5.2-4: Hypothesis Test Scenario for Variances of Normal Distributions



Hypothesis	Test Statistic	Rejection Criteria
H_0: Data are generated from a specific distribution H_1: Data are not generated from the null hypothesis distribution	General Form: $\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$ where, O_i = observed number of occurrences E_i = expected number of occurrences based on assumed distribution Degrees of freedom: n-k-1 where, n = number of intervals of data k = number of intervals to be estimated from the data	$\chi^2_{\alpha,n-k-1}$

Figure A.5.2-5: Hypothesis Test Scenario for an Assumed Distribution



Hypothesis	Test Statistic	Rejection Criteria
H_0: There is no relationship between the variables being tested, i.e., they are statistically independent of each other H_1: There is a real relationship between the variables being tested, i.e., they are not independent.	General Form: $\chi_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$ where, O_i = observed number of occurrences E_i = expected number of occurrences based on assumed distribution Degrees of freedom: Product (r-1)(c01) where, r = # of rows of data c = # of columns of data	$\chi_0^2 > \chi^2_{\alpha,(r-1)(c-1)}$

Figure A.5.2-6: Hypothesis Test Scenario for Independence

The critical values for rejection criteria are generally easiest to determine from look-up tables.

Table A.5.2-4 summarizes a brief example of the hypothesis testing process. For the purposes of this example, assume that a software engineer has written a program and wants to know whether the MTBF of the program is greater than 175 processing hours. From previous programming experience, it is known that the standard deviation of MTBF is 10 processing hours.

Table A.5.2-4: Example of Statistical Hypothesis Testing

Sequence of Steps	Example
1. State the null hypothesis, H_0, and the alternative hypothesis, H_1	H_0: MTBF = 175 CPU processing hours H_1: MTBF > 175 CPU processing hours
2. Specify a significance level, α	Specify Producer's Risk (Type I error) = 0.05
3. Specify a sample size, n	The program was sent to 25 potential users at random
4. Select an appropriate test statistic	Since the standard deviation is known, the appropriate test statistic for this example is (from Figure 3.5.2-3): $$\mathbf{Z}_0 = \frac{\bar{\mathbf{x}} - \mu_0}{\sigma / \sqrt{\mathbf{n}}}$$
5. Define the region of rejection (critical region)	From a table of the standardized normal distribution, the critical value is determined to be: $$\mathbf{Z}_0 > \mathbf{Z}_\alpha = \mathbf{Z}_{0.05} = 1.645$$
6. Compute the value of the statistic and determine whether the null hypothesis should be accepted or rejected.	The observed MTBF from the sample of 25 users was determined to be 182 processing hours. Based on this information, the test statistic is calculated as: $$\mathbf{Z}_0 = \frac{182 - 175}{10 / \sqrt{25}} = 3.50$$ Since 3.50 (Z_0) is greater than 1.645 ($Z_{0.05}$), the null hypothesis of MTBF = 175 processing hours is rejected. The conclusion is that the MTBF of the software program is greater than 175 CPU processing hours. **NOTE: There is no claim as to what the true MTBF of the software program is and a 5% risk that the conclusion is wrong!**

For More Information:

1. Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008
2. Madsen, R.W.; Moeschberger, M.L., "Statistical Concepts with Applications to Business and Economics", Prentice-Hall, 1980, ISBN 0138448787
3. Montgomery, D.C., "Introduction to Statistical Quality Control – Second Edition", John Wiley & Sons, 1991, ISBN 047151988X
4. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587
5. http://simon.cs.vt.edu/SoSci/converted/Hypoth_I/

Appendix A.5.2.1: Hypothesis Testing for Reliability Acceptance

Hypothesis testing for reliability acceptance involves a decision as to whether the reliability observed during a controlled test satisfies a specified minimum level of required reliability.

As an example, consider a system that is being tested in an environment with a constant operational profile. The system is restarted when a failure occurs, and no redesign is performed to correct experienced faults. This type of test can be modeled as a Homogeneous Poisson Process, i.e., the times between failures are independent and identically-distributed random variables from an exponential distribution. We want to decide between two values of the population mean for the exponential distribution.

The hypothesis to be tested is:

<u>Null Hypothesis</u>: $\boldsymbol{H_0: \theta = \theta_0}$ **(good system)**

where,

θ = mean of the distribution of times between failures

θ_0 = the desired/required MTBF for a good system

<u>Alternative Hypothesis</u>: $\boldsymbol{H_1: \theta = \theta_1}$**, where** $\boldsymbol{\theta_1 < \theta_0}$ **(bad system)**

where,

θ_1 = MTBF for a bad system

The appropriate Chi-square percentile, defined as $u_{1-\alpha,2n}$, is determined from a Chi-square table look up using the desired level of confidence, γ (or a desired level of Producer's Risk, α, where $\gamma = 1-\alpha$):

$$P\{U < \chi^2_{1-\alpha,2n}\} = \gamma$$

where,

U = the random variable from a Chi-square distribution

$\chi^2_{1-\alpha,2n}$ = the look-up value from a Chi-square distribution table at the 100 $(1-\alpha)^{th}$ percentile for "2n" degrees of freedom

α = the Producer's/Supplier's risk (Type I error)

n = the number of faults experienced during the test

γ = the probability that the true MTBF is above the value for a "bad" system

The lower 100 $(1-\alpha)$% confidence bound on the observed MTBF is calculated using the formula:

$$\theta_L = \frac{2t}{\chi^2_{1-\alpha,2n}}$$

where,

t = total time on test

$\chi^2_{1-\alpha,2n}$ = Chi-square percentile

α = significance level = Producer's risk (Type I error)

n = number of observed faults

If the calculated confidence bound is less than θ_1, then the null hypothesis that the system is good ($\theta = \theta_0$) should be rejected in favor of the alternative hypothesis ($\theta = \theta_1$, where $\theta_1 < \theta_0$). In this context, the significance level, α, is the probability of making an incorrect decision by rejecting a good system.

As an example of a reliability acceptance test requirement, suppose that a good system is defined to have a MTBF of θ_0 = 72 hours, and a bad system is defined to have a MTBF of θ_1 = 24 hours. During the course of the reliability acceptance test, failures were observed at the times indicated in Table A.5.2.1-1. The steps for analyzing this hypothesis are described in Table A.5.2.1-2. A portion of a Chi-square table is reproduced in Table A.5.2.1-3.

Table A.5.2.1-1: Failure Times for Reliability Acceptance Test Example

Failure Number	Time Between Failure (Hours)
1	11.52
2	34.56
3	24.96
4	44.16
5	26.88
6	43.20
7	22.92
8	15.60

Table A.5.2.1-2: Steps for Reliability Acceptance Test Example

Step	Example
1. Determine times between successive failures, $t_1, t_2, t_3, \ldots, t_n$	See Table 3.5.2.1-1
2. Calculate the total time on test, t: $$\mathbf{t} = \sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{t}_{\mathbf{i}}$$	The total time on test is: $$\mathbf{t} = \sum_{\mathbf{i}=1}^{8} \mathbf{t}_{\mathbf{i}} = 223.80 \text{ hours}$$
3. Find the appropriate Chi-square percentile, $\chi_{1-\alpha,2n}$, based on the required significance level, α	Assuming a significance level of 0.10 (10% Producer's/Supplier's risk) and using a Chi-Square table with 2n = (2)(8) = 16 degrees of freedom: $$\chi^2_{1-0.10,16} = \chi^2_{0.90,16} = 23.54$$
4. Calculate the lower confidence bound on MTBF: $$\theta_L = \frac{2\mathbf{t}}{\mathbf{u}_{1-\alpha,2\mathbf{n}}}$$	The calculation of the lower 90% confidence bound from the measured data is: $$\theta_L = \frac{(2)(223.80)}{23.54} = 19.01 \text{hours}$$
5. Reject the null hypothesis if Step 4 confidence bound is $\leq \theta_1$	The calculated lower bound of the MTBF (19 hours) is lower than what is considered a "bad" system (24 hours). The null hypothesis is rejected, i.e., the system is considered "bad" at a producer's risk of 10% (a 10% probability that a "good" system is rejected as "bad").

Table A.5.2.1-3: Partial Chi-Square Distribution Table

df \ P	0.500	0.750	0.900	0.950	0.975	0.990	0.995	0.999
1	0.4549	1.323	2.706	3.841	5.024	6.635	7.879	10.83
2	1.3860	2.773	4.605	5.991	7.378	9.210	10.600	13.82
3	2.3660	4.108	6.251	7.815	9.348	11.340	12.840	16.27
4	3.3570	5.385	7.779	9.488	11.140	13.280	14.860	18.47
5	4.3510	6.626	9.236	11.070	12.830	15.090	16.750	20.52
6	5.3480	7.841	10.640	12.590	14.450	16.810	18.550	22.46
7	6.3460	9.037	12.020	14.070	16.010	18.480	20.280	24.32
8	7.3440	10.220	13.360	15.510	17.530	20.090	21.960	26.12
9	8.3430	11.390	14.680	16.920	19.020	21.670	23.590	27.88
10	9.3420	12.550	15.990	18.310	20.480	23.210	25.190	29.59
11	10.3400	13.700	17.280	19.680	21.920	24.720	26.760	31.26
12	11.3400	14.850	18.550	21.030	23.340	26.220	28.300	32.91
13	12.3400	15.980	19.810	22.360	24.740	27.690	29.820	34.53
14	13.3400	17.120	21.060	23.680	26.120	29.140	31.320	36.12
15	14.3400	18.250	22.310	25.000	27.490	30.580	32.800	37.70
16	15.3400	19.370	23.540	26.300	28.850	32.000	34.270	39.25
17	16.3400	20.490	24.770	27.590	30.190	33.410	35.720	40.79
18	17.3400	21.600	25.990	28.870	31.530	34.810	37.160	42.31
19	18.3400	22.720	27.200	30.140	32.850	36.190	38.580	43.82
20	19.3400	23.830	28.410	31.410	34.170	37.570	40.000	45.32
21	20.3400	24.930	29.620	32.670	35.480	38.930	41.400	46.80
22	21.3400	26.040	30.810	33.920	36.780	40.290	42.800	48.27
23	22.3400	27.140	32.010	35.170	38.080	41.640	44.180	49.73
24	23.3400	28.240	33.200	36.420	39.360	42.980	45.560	51.18
25	24.3400	29.340	34.380	37.650	40.650	44.310	46.930	52.62
26	25.3400	30.430	35.560	38.890	41.920	45.640	48.290	54.05
27	26.3400	31.530	36.740	40.110	43.190	46.960	49.640	55.48
28	27.3400	32.620	37.920	41.340	44.460	48.280	50.990	56.89
29	28.3400	33.710	39.090	42.560	45.720	49.590	52.340	58.30
30	29.3400	34.800	40.260	43.770	46.980	50.890	53.670	59.70
31	30.3381	35.911	41.540	45.102	48.235	52.354	55.092	61.32
32	31.3380	36.997	42.705	46.312	49.484	53.650	56.417	62.71
33	32.3378	38.082	43.867	47.520	50.728	54.941	57.737	64.09
34	33.3377	39.166	45.027	48.724	51.969	56.228	59.053	65.47
35	34.3376	40.248	46.185	49.925	53.207	57.510	60.364	66.84
36	35.3374	41.330	47.340	51.123	54.441	58.788	61.670	68.21
37	36.3373	42.410	48.493	52.318	55.671	60.063	62.972	69.57
38	37.3372	43.489	49.644	53.511	56.899	61.334	64.270	70.92
39	38.3371	44.567	50.792	54.701	58.123	62.601	65.565	72.27
40	39.3370	45.644	51.939	55.889	59.345	63.865	66.855	73.62
41	40.3369	46.720	53.084	57.074	60.564	65.125	68.142	74.96
42	41.3369	47.795	54.228	58.258	61.780	66.383	69.425	76.30
43	42.3368	48.869	55.369	59.438	62.994	67.637	70.705	77.64
44	43.3367	49.943	56.509	60.617	64.205	68.888	71.982	78.97
45	44.3366	51.015	57.647	61.794	65.414	70.137	73.255	80.30
46	45.3365	52.087	58.784	62.969	66.620	71.383	74.526	81.62
47	46.3365	53.158	59.919	64.141	67.824	72.626	75.794	82.94
48	47.3364	54.228	61.053	65.312	69.026	73.866	77.058	84.26
49	48.3364	55.297	62.186	66.482	70.226	75.104	78.320	85.57
50	49.3363	56.366	63.317	67.649	71.424	76.339	79.580	86.88

For More Information:

1. Nelson, W., “Applied Life Data Analysis”, John Wiley & Sons, 1982, ISBN 0471094587

Appendix A.5.2.2: Hypothesis Testing for Reliability Growth

Reliability growth, in either the positive or negative direction, can occur throughout the system life cycle as analyses and testing is performed to uncover deficiencies and verify that corrective actions have been identified, implemented, and proven effective to prevent reoccurrence of those deficiencies after the system is delivered to the user.

Methods for formal reliability growth testing will be covered in more detail in a later section of this Handbook. This section deals with hypothesis testing that can be performed on observed data to statistically determine whether a failure intensity function (i.e., failure rate) is constant, increasing, or decreasing. The hypothesis to be tested is:

Null Hypothesis — **H_0: The observed data are generated from a homogeneous Poisson process (HPP).**

By definition, the failure intensity of a HPP is constant.

Alternative Hypothesis — **H_1: The failure intensity function is either monotonically decreasing or monotonically increasing (nonhomogeneous Poisson process (NHPP)**

The failure rate is either decreasing (the "infant mortality" portion of the reliability bathtub curve) or increasing (the "wear-out" portion of the bathtub curve)

A test based on the Laplace statistic can be used to statistically accept or reject the null hypothesis. Under the null hypothesis, the Laplace statistic is normally distributed, with a mean of zero and a standard deviation of one (i.e., the standard normal distribution). Positive values of the Laplace statistic indicate an increasing failure rate (wear-out). Negative values of the Laplace statistic indicate a decreasing failure rate (infant mortality). When the Laplace statistic equals zero, the failure rate is constant.

In order to illustrate an example of this hypothesis test, Table A.5.2.2-1 contains data that represents observed times between failures for 10 failures of a system. The steps taken to calculate and apply the Laplace statistic to accept or reject the null hypothesis are described in Table A.5.2.2-2.

Table A.5.2.2-1: Example Data for Reliability Growth Hypothesis Test

Failure Number	Inter-Arrival Hours
1	0.9105
2	0.8151
3	0.2360
4	1.6250
5	0.0629
6	3.3390
7	4.1900
8	4.9830
9	4.5260
10	5.4390

Table A.5.2.2-2: Steps for Reliability Growth Example

<table>
<tr><th>Step</th><th>Example</th></tr>
<tr><td>1. Determine times between successive failures, t_1, t_2, t_3, … , t_n</td><td>See Table A.5.2.2-1</td></tr>
<tr><td>2. Calculate the cumulative times to failure (TTF):
$t_i = x_1 + x_2 + x_3 + \ldots + x_i$, for i = 1, 2, 3, …, n

where,
t_i = cumulative time to the i^{th} failure
x_i = inter-arrival time to the i^{th} failure</td><td>For this example, the calculated cumulative times to failure are:
<table>
<tr><th>Failure Number</th><th>Cum. TTF</th><th>Failure Number</th><th>Cum. TTF</th></tr>
<tr><td>1</td><td>0.9105</td><td>6</td><td>6.9885</td></tr>
<tr><td>2</td><td>1.7256</td><td>7</td><td>11.1785</td></tr>
<tr><td>3</td><td>1.9616</td><td>8</td><td>16.1615</td></tr>
<tr><td>4</td><td>3.5866</td><td>9</td><td>20.6875</td></tr>
<tr><td>5</td><td>3.6495</td><td>10</td><td>26.1265</td></tr>
</table></td></tr>
<tr><td>3. Calculate the running sum of the cumulative times to failure (TTF):
$$\mathbf{t_n} = \sum_{\mathbf{i}=1}^{\mathbf{n}-1} \mathbf{t_i}$$
where,
t_n = running sum of cumulative times to failure
t_i = cumulative time to the i^{th} failure</td><td>For this example, the calculated running sum of cumulative times to failure are:
<table>
<tr><th>Failure Number</th><th>Running Sum</th><th>Failure Number</th><th>Running Sum</th></tr>
<tr><td>1</td><td>0.9105</td><td>6</td><td>18.8223</td></tr>
<tr><td>2</td><td>2.6361</td><td>7</td><td>30.0008</td></tr>
<tr><td>3</td><td>4.5977</td><td>8</td><td>46.1623</td></tr>
<tr><td>4</td><td>8.1843</td><td>9</td><td>66.8948</td></tr>
<tr><td>5</td><td>11.8338</td><td>10</td><td></td></tr>
</table></td></tr>
<tr><td>4. Find the critical value for the standard normal percentile, $z_{(1-\alpha)}$ for a one-sided test or $z_{(1+(1-\alpha))/2}$ for a two-sided test, based on the required significance level, α</td><td>Assuming a significance level of 0.05 and using a standard normal table for a two-sided test:
$$\mathbf{z}_{(1+(1-\boldsymbol{\alpha}))/2} = \mathbf{z}_{0.975} = 1.960$$</td></tr>
<tr><td>5. Calculate the Laplace statistic for individual failures, and for the overall sample, using the formula:
$$\mathbf{u}(\mathbf{n}) = \frac{\frac{1}{\mathbf{n}-1}\left(\mathbf{t}_1 + \mathbf{t}_2 + \cdots + \mathbf{t}_{\mathbf{n}-1}\right) - \frac{\mathbf{t_n}}{2}}{\mathbf{t_n}\sqrt{\frac{1}{12(\mathbf{n}-1)}}}$$</td><td>The calculated Laplace statistic at the time of the 10^{th} failure is:
$$\mathbf{u}(\mathbf{n}) = \frac{\frac{1}{9}(66.8948) - \frac{26.1265}{2}}{26.1265\sqrt{\frac{1}{12(9)}}} = \frac{-5.630}{2.514} = -2.239$$
The Laplace statistics for the 1^{st} through 9^{th} failures are calculated and tabulated below.
<table>
<tr><th>Failure Number</th><th>Laplace Statistic</th><th>Failure Number</th><th>Laplace Statistic</th></tr>
<tr><td>1</td><td></td><td>6</td><td>-1.250</td></tr>
<tr><td>2</td><td>0.0958</td><td>7</td><td>-1.861</td></tr>
<tr><td>3</td><td>0.8420</td><td>8</td><td>-2.152</td></tr>
<tr><td>4</td><td>-0.4360</td><td>9</td><td>-2.166</td></tr>
<tr><td>5</td><td>0.4200</td><td>10</td><td>-2.239</td></tr>
</table></td></tr>
<tr><td>6. Reject the null hypothesis if the absolute value of the Laplace statistic exceeds the standard normal percentile at the desired significance level.</td><td>The absolute value of the Laplace statistic at the 10^{th} failure is $\lvert -2.239 \rvert = 2.239$, which is greater than the critical value from the standard normal distribution, $\mathbf{z}_{0.975} = 1.960$ for a two-sided test (Step 4). The null hypothesis that the data is generated from a process having a constant failure rate is rejected. The Laplace statistic, since it is negative, indicates that the observed data is from a process having a decreasing failure rate (positive reliability growth). Since the Laplace statistic can be recalculated at each failure, the process can be continually monitored for growth.

Upon closer observation of the table from Step 5, the Laplace statistic could indicate either the successful implementation of a corrective action following failure number 5, or simply statistical variation with a small sample size (weak power in the test).</td></tr>
</table>

Appendix A.5.2.3: Chi-Square Goodness-of-Fit Test

In the statistical analysis of failure data it is common practice to assume that observed failure times follow a specific failure distribution type. This assumption may be based on historical data, or simply on (informed) engineering judgment.

The Chi-square goodness-of-fit test (where Chi-square is represented by the symbol χ^2) is used to test the validity of any assumed discrete or continuous distribution (i.e., it is "distribution-free") when the values of its random variables fall into discrete categories. In other words, the test is used to determine if empirical data disproves the hypothesis of fit to the assumed distribution.

The test is not directly dependent on sample size but, rather, it is dependent on the number of intervals into which the scale of failure times is divided. The only restriction is that all expected values should be greater than one and at least 80% of the expected values should be greater than five. Adjacent categories should be combined if these conditions are not met. The Chi-square test is, therefore, best used when there are a relatively large number of observed failures.

The Kolmogorov-Smirnov goodness-of-fit test discussed in Appendix A.5.2.4 is preferred over the Chi-square if individual failure times are known, but the Chi-square test has two distinct advantages over the Kolmogorov-Smirnov test:

- Chi-square can be partitioned and added
- Chi-square can be applied to discrete populations

As an example, consider whether the observed number of failures in successive days of testing is from a Poisson distribution:

Null Hypothesis **H_0: The data are generated from a Poisson distribution with mean, μ**

where the mean, μ, is estimated by the sample mean, $\hat{\mu}$, as:

$$\hat{\mu} = \frac{1}{n}\left(x_1 + x_2 + \ldots + x_n\right)$$

and x_1, x_2, …, x_n are number of failures observed in successive days

Alternative Hypothesis **H_1: The data are not generated from a Poisson distribution**

The data that will be used to test this hypothesis is presented in Table A.5.2.3-1, which presents the number of failures experienced per day over the period of a twenty-day test. The steps to be taken in performing the Chi-square goodness-of-fit test and determining whether to accept or reject the null hypothesis are provided in Table A.5.2.3-2.

Table A.5.2.3-1: Example Data for Chi-Square Goodness-of-Fit Test

Day	Failures	Day	Failures
1	2	11	1
2	1	12	1
3	1	13	2
4	3	14	1
5	1	15	2
6	2	16	0
7	0	17	0
8	0	18	2
9	0	19	1
10	1	20	1

Table A.5.2.3-2: Steps for Chi-Square Goodness-of-Fit Example

Step	Example
1. Determine the underlying distribution to be tested	The null hypothesis has been set up to test the Poisson distribution
2. Determine a level of significance, α, as the risk of rejecting the underlying distribution if, in fact, it is the real distribution	Define α = 0.10 (Type I error; significance level; a 10% probability of rejecting the hypothesis that the data comes from a Poisson distribution when the data does, in fact, come from the Poisson distribution)
3. Divide the scale into "k" intervals or categories, where the intervals/categories may represent time, distance, volume, number of failures, etc.	Using the data in Table A.5.2.3-1, divide the scale (number of failures) into 3 categories: • Category 0 = number of days that no failures were experienced • Category 1 = number of days that exactly one failure was experienced • Category >1 = number of days that more than one failure was experienced (there was only one day when more than 2 failures were experienced, so Day 4 is combined with Days 1, 6, 13, 15, and 18)
4. Determine the number of sample observations falling within each defined interval or category	Using the data in Table A.5.2.3-1, the number of observations in each category are: • Category 0 = 5 days with no failures (**O_0**) • Category 1 = 9 days with exactly one failure (**O_1**) • Category >1 = 6 days with more than one failure (**O_2**)
5. Using the assumed underlying distribution, calculate the expected number of observations in each interval. For the Poisson distribution: $\mathbf{E_i} = \mathbf{n}\frac{\hat{\mu}^{\mathbf{i}}}{\mathbf{i!}}\mathbf{e}^{-\hat{\mu}}$ If an exponential distribution was assumed: $\mathbf{E_i} = \mathbf{n}\left[\mathbf{e}^{\frac{-\mathbf{L_i}}{\hat{\theta}}} - \mathbf{e}^{\frac{-\mathbf{U_i}}{\hat{\theta}}}\right]$ where L_i and U_i represent the lower and upper values of the interval for which the expected number of observations is being calculated. In both cases, the expected value of E_{m-1} (number of days with m-1 failures or more) = $n-(E_0 + E_1 + \ldots + E_{m-2})$	The sample mean is calculated as 22 failures/20 days, resulting in 1.10 failures per day. The total observed frequency, n, of days with failures is 5 + 9 + 6 = 20 There are a total of 3 categories, so $m = 3$, $E_{m-1} = E_2$, and $E_{m-2} = E_1$ For the Poisson distribution, the number of expected observations in each category are: • Category 0 = $\mathbf{E}_0 = (20)\frac{(1.1)^0}{0!}\mathbf{e}^{-1.1} = 6.6574$ • Category 1 = $\mathbf{E}_1 = (20)\frac{(1.1)^1}{1!}\mathbf{e}^{-1.1} = 7.3232$ • Category >1 = $\mathbf{E}_2 = 20-(6.6574+7.3232) = 6.0194$

Table A.5.2.3-2: Steps for Chi-Square Goodness-of-Fit Example (continued)

<table>
<tr><th>Step</th><th>Example</th></tr>
<tr>
<td>6. Calculate the value of the observed Chi-square statistic:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

where,
O_i = number of sample observations in interval "i"
E_i = expected number of observations in interval "i"
k = number of intervals</td>
<td>The calculated values for the example are:
<table>
<tr><th>Category</th><th>Observed (O_i)</th><th>Expected (E_i)</th><th>$\frac{(O_i - E_i)^2}{E_1}$</th></tr>
<tr><td>0</td><td>5</td><td>6.6574</td><td>0.41262</td></tr>
<tr><td>1</td><td>9</td><td>7.3232</td><td>0.38394</td></tr>
<tr><td>>1</td><td>6</td><td>6.0194</td><td>0.00006</td></tr>
<tr><td></td><td></td><td>$\chi^2 =$</td><td>0.79662</td></tr>
</table>
</td>
</tr>
<tr>
<td>7. Determine the critical value of the Chi-square statistic from a look-up table:

$$\chi^2_{1-\alpha, k-w-1}$$

where,
α = desired significance level (Type I error)
k = number of intervals
w = number of parameters estimated from the data</td>
<td>For this example, α = 0.10, k = 3, and the number of parameters estimated from the data, w, is 1 (the sample mean). The Chi-square critical value (from Table 3.5.2.3-3) is.

$$\chi^2_{1-0.10,3-1-1} = \chi^2_{0.90,1} = 2.706$$</td>
</tr>
<tr>
<td>8. Reject the distribution under test if:

$$\sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} > \chi^2_{1-\alpha, k-w-1}$$

Otherwise, there is insufficient evidence to reject the assumed underlying distribution</td>
<td>For this example, 0.79662 is not greater than 2.70554, so there is insufficient statistical evidence to reject the null hypothesis that the example data come from a Poisson distribution</td>
</tr>
</table>

Table A.5.2.3-3: Partial Chi-Square Distribution Table

df	P	0.500	0.750	0.900	0.950	0.975	0.990	0.995	0.999
1		0.4549	1.323	2.706	3.841	5.024	6.635	7.879	10.83
2		1.3860	2.773	4.605	5.991	7.378	9.210	10.600	13.82
3		2.3660	4.108	6.251	7.815	9.348	11.340	12.840	16.27
4		3.3570	5.385	7.779	9.488	11.140	13.280	14.860	18.47
5		4.3510	6.626	9.236	11.070	12.830	15.090	16.750	20.52

For More Information:

1. Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008
2. MIL-HDBK-338, "Electronic Design Handbook", Section 8.3.2.6.2
3. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN 0471094587
4. http://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm

Appendix A.5.2.4: Kolmogorov-Smirnov Goodness-of-Fit Test

The Kolmogorov-Smirnov (K-S) goodness-of-fit test (sometimes referred to as the "d" test) is based, like the Chi-square test, on the fact that the observed cumulative distribution of sample data is expected to be fairly close to the true statistical distribution of the population. For this test, the goodness-of-fit is measured by finding the point at which the sample and the population are farthest apart and comparing this distance with an entry in a Kolmogorov-Smirnov table of critical values. Comparing this distance with the critical value will indicate the likelihood of such a distance occurring. If the distance is excessive, the chance that the observations actually come from a population with the specified distribution is very small (reject the null hypothesis).

The process begins, once again, with a suggestion derived from either historical data or engineering judgment that failure times of interest are from a specific failure distribution. Like the Chi-square, the K-S goodness-of-fit test is distribution-free, i.e., it can be used regardless of the failure distribution that the data are assumed to follow.

The discriminating capability of the K-S test is dependent on sample size. The larger the sample size, the more reliable the results. When large sample sizes are available, the Chi-square test tends to be more powerful, but at the expense of increased manipulation of the sample data. For small sample sizes, the K-S test only provides limited information, but still represents a better choice than the Chi-square test. In the strictest sense, the K-S goodness-of-fit test does require prior knowledge of the population parameters (the Chi-square test does not). If parameters need to be estimated from the sample, then the exact error risks associated with K-S test results are unknown.

The distinct advantages of the Kolmogorov-Smirnov goodness-of-fit test over the Chi-square test are:

- It can be used to test for deviations in a given direction, while the Chi-square test can be used only for a two-sided test
- It uses ungrouped data, so that every observation represents a point of comparison. The Chi-square test requires its data to be grouped into cells representing an arbitrary choice of interval, size, and selection of a starting point. The Chi-square test also requires minimum expected frequency values.
- It can be used in a sequential test where data become available from the smallest to the largest elapsed period. Computations need only be continued up to the point at which rejection of the null hypothesis occurs.

As an example, a null hypothesis to be tested is whether observed inter-arrival failure times are from an exponential distribution.

<u>Null Hypothesis</u> **H_0:** $\mathbf{F}_0(\mathbf{t}) = 1 - \mathbf{e}^{-\lambda \mathbf{t}}, \quad \mathbf{t} > 0$

where,

$\mathbf{F}_0(\mathbf{t})$ = the CDF of the time between failure

$\hat{\boldsymbol{\lambda}}$ = the failure rate estimated from the data:

$$\hat{\boldsymbol{\lambda}} = \mathbf{n} / \left(\mathbf{t}_1 + \mathbf{t}_2 + \mathbf{t}_3 + \ldots + \mathbf{t}_\mathbf{n}\right)$$

where,

$t_1, t_2, \ldots, t_n$ are times between successive failures

Alternative Hypothesis $\mathbf{H_1}$: $F_0(t) \neq 1 - e^{-\lambda t}$

The sorted inter-arrival failure times define an empirical CDF, $S[t_i]$, where the empirical CDF is the proportion of observed inter-failure times less than or equal to the argument:

$$S[t_i] = i/n$$

where t_i is the i^{th} order statistic for the inter-failure times and "n" is the number of observed failures. The Kolmogorov-Smirnov statistic, "d", is the maximum distance between the empirical CDF and the CDF under the null hypothesis:

$$d = \text{maximum } |F_0[t_i] - S[t_i]|$$

The raw data that will be used to illustrate this example is presented in Table A.5.2.4-1. Time to Failure is in system operating hours. The steps involved in performing the K-S test analytically are illustrated in Table A.5.2.4-2.

Table A.5.2.4-1: Example Data for Kolmogorov-Smirnov Goodness-of-Fit Test

Failure No.	Time	Failure No.	Time
1	1.1060	**9**	1.1900
2	1.8460	**10**	1.1950
3	0.2692	**11**	0.8310
4	0.7225	**12**	0.6560
5	1.2140	**13**	0.4366
6	0.7560	**14**	0.8345
7	0.4773	**15**	0.6999
8	1.2000		

Table A.5.2.4-2: Steps for Kolmogorov-Smirnov Goodness-of-Fit Example

<table>
<tr><th>Step</th><th>Example</th></tr>
<tr><td>1. Observe, record, and rank inter-arrival failure times (in increasing order)</td><td>The ranked failure times for the observed failures are:
<table>
<tr><th>Index No.</th><th>Time</th><th>Index No.</th><th>Time</th></tr>
<tr><td>t_1</td><td>0.2692</td><td>t_9</td><td>0.8345</td></tr>
<tr><td>t_2</td><td>0.4366</td><td>t_{10}</td><td>1.1060</td></tr>
<tr><td>t_3</td><td>0.4773</td><td>t_{11}</td><td>1.1900</td></tr>
<tr><td>t_4</td><td>0.6560</td><td>t_{12}</td><td>1.1950</td></tr>
<tr><td>t_5</td><td>0.6999</td><td>t_{13}</td><td>1.2000</td></tr>
<tr><td>t_6</td><td>0.7225</td><td>t_{14}</td><td>1.2140</td></tr>
<tr><td>t_7</td><td>0.7560</td><td>t_{15}</td><td>1.8460</td></tr>
<tr><td>t_8</td><td>0.8310</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>2. Determine a level of significance, α, as the risk of rejecting the underlying distribution if, in fact, it is the real distribution</td><td>Define $\alpha = 0.05$ (Type I error; significance level; a 5% probability of rejecting the hypothesis that the data comes from an exponential distribution when the data does, in fact, come from the exponential distribution)</td></tr>
<tr><td>3. Estimate the parameters of the assumed distribution from the observed data</td><td>The estimated failure rate from the data is:
$\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} t_i} = \frac{15}{13.434} = 1.11657 \text{ failures/processor hour}$</td></tr>
</table>

Table A.5.2.4-2: Steps for Kolmogorov-Smirnov Goodness-of-Fit Example (continued)

<table>
<tr><th>Step</th><th>Example</th></tr>
<tr><td>4. Calculate the probability of failure, $F_0(t_i)$, for each observation from the cumulative failure function for the assumed distribution</td><td>Using the ranked data in Step 1 above, the individual probability of each failure is calculated using:

$$F_0(t) = 1 - e^{-\lambda t}, \quad t > 0$$

<table>
<tr><th>Index No.</th><th>$F_0(t_i)$</th><th>Index No.</th><th>$F_0(t_i)$</th></tr>
<tr><td>t_1</td><td>0.2596</td><td>t_9</td><td>0.6061</td></tr>
<tr><td>t_2</td><td>0.3858</td><td>t_{10}</td><td>0.7091</td></tr>
<tr><td>t_3</td><td>0.4131</td><td>t_{11}</td><td>0.7352</td></tr>
<tr><td>t_4</td><td>0.5193</td><td>t_{12}</td><td>0.7367</td></tr>
<tr><td>t_5</td><td>0.5423</td><td>t_{13}</td><td>0.7381</td></tr>
<tr><td>t_6</td><td>0.5537</td><td>t_{14}</td><td>0.7422</td></tr>
<tr><td>t_7</td><td>0.5701</td><td>t_{15}</td><td>0.8727</td></tr>
<tr><td>t_8</td><td>0.6046</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>5. Calculate the Kolmogorov-Smirnov paired statistics for each indexed failure using the formulae:

For each t_i :

$$d_1 = \left(\frac{i}{n}\right) - F_0(t_i)$$

$$d_2 = F_0(t_i) - \frac{(i-1)}{n}$$

Then determine the K-S statistic, "d", as:

$$d = \underset{i=1,2,\ldots,n}{\text{maximum}} \left\{ \frac{i}{n} - F_0[x_i], F_0[x_i] - \frac{i-1}{n} \right\}$$
</td><td>For n =15, the K-S paired statistics for each "t_i" are calculated as:

<table>
<tr><th>Index No.</th><th>$F_0(t_i)$</th><th>d_1</th><th>d_2</th></tr>
<tr><td>t_1</td><td>0.2596</td><td>-0.1930</td><td>0.2596</td></tr>
<tr><td>t_2</td><td>0.3858</td><td>-0.2525</td><td>0.3192</td></tr>
<tr><td>t_3</td><td>0.4131</td><td>-0.2131</td><td>0.2798</td></tr>
<tr><td>t_4</td><td>0.5193</td><td>-0.2526</td><td>**0.3193**</td></tr>
<tr><td>t_5</td><td>0.5423</td><td>-0.2089</td><td>0.2756</td></tr>
<tr><td>t_6</td><td>0.5537</td><td>-0.1537</td><td>0.2203</td></tr>
<tr><td>t_7</td><td>0.5701</td><td>-0.1034</td><td>0.1701</td></tr>
<tr><td>t_8</td><td>0.6046</td><td>-0.07127</td><td>0.1379</td></tr>
<tr><td>t_9</td><td>0.6061</td><td>-0.00615</td><td>0.07281</td></tr>
<tr><td>t_{10}</td><td>0.7091</td><td>-0.04248</td><td>0.1091</td></tr>
<tr><td>t_{11}</td><td>0.7352</td><td>-0.00185</td><td>0.06852</td></tr>
<tr><td>t_{12}</td><td>0.7367</td><td>0.06334</td><td>0.003324</td></tr>
<tr><td>t_{13}</td><td>0.7381</td><td>0.1285</td><td>-0.06188</td></tr>
<tr><td>t_{14}</td><td>0.7422</td><td>0.1911</td><td>-0.1245</td></tr>
<tr><td>t_{15}</td><td>0.8727</td><td>0.1273</td><td>-0.06064</td></tr>
</table>

From the above table, the maximum value of the K-S statistic is 0.3193</td></tr>
<tr><td>9. Determine the critical value of the K-S statistic from an appropriate table based on the sample size, "n", and the desired significance level, α</td><td>**Note: In this example, it was necessary to estimate the failure rate parameter, $\hat{\lambda}$. As a result, since a significance level of 5% was specified, the critical K-S statistic value is taken from the 10% column of Table 3.4.2-3 for a sample size of n=15. Similarly, a specified α of 1% would use the 5% column of the table, and a specified α of 10% would use the 20% column of the table. If the true population failure rate, λ , was known, then there would be direct correlation between the specified α and the table lookup value.**

The critical value of the K-S statistic from Table A.5.2.4-3 is 0.304</td></tr>
<tr><td>10. Compare the largest value of the observed K-S statistic (Step 5) with the critical value of the K-S statistic (Step 7) to test for goodness-of-fit. If the observed statistic is not larger than the critical value, then the null hypothesis (failure times are from the assumed distribution) is accepted.</td><td>The statistic calculated from the data (0.3913) is larger than the critical value (0.304). Thus, one should conclude that these inter-arrival failure times are generated from a distribution other than an exponential distribution.</td></tr>
</table>

Table A.5.2.4-3: Partial Kolmogorov-Smirnov Significance Levels

Sample Size	Significance Level			
	20%	**10%**	**5%**	**1%**
1	0.900	0.950	0.975	0.995
2	0.684	0.776	0.842	0.929
3	0.565	0.642	0.708	0.828
4	0.494	0.564	0.624	0.733
5	0.446	0.510	0.565	0.669
6	0.410	0.470	0.521	0.618
7	0.381	0.438	0.486	0.577
8	0.358	0.411	0.457	0.543
9	0.339	0.388	0.432	0.514
10	0.322	0.368	0.410	0.490
11	0.307	0.352	0.391	0.468
12	0.295	0.338	0.375	0.450
13	0.284	0.325	0.361	0.433
14	0.274	0.314	0.349	0.418
15	0.266	**0.304**	0.338	0.404

For More Information:

1. Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008
2. MIL-HDBK-338, "Electronic Design Handbook", Section 8.3.2.6.1
3. http://www.physics.csbsju.edu/stats/KS-test.html
4. http://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm

Appendix A.5.3: Parameter Estimation

Statistics involve drawing inferences from realizations of random variables, such as observed failure times. Typical inferences consist of point and interval estimates of distribution parameters and decisions in statistical hypothesis testing.

Parameter estimation provides a means for the effective use of data to aid in mathematical modeling and the estimation of constants appearing in those models. The constants that appear in distribution functions (e.g., "p" in the binomial distribution; "λ" in the Poisson distribution; "μ" and "σ" in the normal distribution; "λ" or "θ" in the exponential distribution; and "α" and "β" in the Weibull distribution) are called parameters. The true value of the parameters from a given distribution may not be known or measurable, so it becomes more practical to obtain approximate or estimated values of these parameters from a sample of data. In the larger context, parameter estimation is typically applied to one of the following scenarios:

- **Criterion:** the choice of the best function to optimize (minimize or maximize)
- **Estimation:** the optimization of a chosen function
- **Design:** optimal design to obtain the best parameter estimates
- **Modeling:** the determination of the mathematical model that best describes the system from which data are measured

Point estimation is frequently used in reliability analysis to quantify parameters dealing with fault detection coverage resulting from fault injections and the estimation of mean time to failure (MTTF) or failure rates being experienced in the field.

Formally, a statistic, Y, is a function of random variables that does not depend on any unknown parameter:

$$Y = u(X_1, \ldots, X_n)$$

Let "θ" denote the parameter to be estimated. Consider functions $w(Y)$ of the statistic, which might serve as point estimates of the parameter. Since $w(Y)$ is a random variable, it has a probability distribution. Statisticians have defined certain properties for assessing the quality of estimators. These properties are defined in terms of this probability distribution.

A loss function, $L[\theta, w(Y)]$, assigns a number to the deviation between a parameter and an estimator. A typical loss function is the square of the difference:

$$L[\theta, w(Y)] = [\theta - w(Y)]^2$$

The risk function is the expected value of the loss function:

$$R(\theta, w) = E\{L[\theta, w(Y)]\}$$

An unbiased estimator that minimizes the risk function for the above loss function is a minimum variance unbiased estimator. An estimator that minimizes this risk function uniformly in θ is called a minimum mean squared estimator. Table A.5.3-1 summarizes the terms most commonly used in parameter estimation.

Table A.5.3-1: Terminology Used In Parameter Estimation

Term	Definition
Confidence Level	The theoretical percentage (or probability) of an interval estimate containing the parameter, and in which the endpoints of the interval are constructed from sample data
Consistent Estimator	The estimate converges to the true value of the parameter as the sample size increases to infinity
Estimator	A function of a statistic used to estimate a parameter in a probability model
Interval Estimator	Estimates of the endpoints of an interval around a parameter
Likelihood	The probability weight for given values of parameters at observed data points
Loss Function	A function that provides a measure of the distance between a parameter value and its estimator
Maximum Likelihood Estimate	An estimate that maximizes the probability that given parameter values will occur at observed data points
Minimum Mean Squared Estimate	An estimator that uniformly minimizes the expected value of the square of the difference between a parameter and an estimator
Minimum Variance Unbiased Estimator	Of all unbiased estimators, none has a smaller variance. Sometimes called a "best" estimator
Risk Function	The mathematical expectation of the loss function
Sample Size	The number of random variables from which a statistic is calculated
Unbiased Estimator	An estimator with a mathematical expectation equal to the parameter being estimated

Table A.5.3-2 provides an overview of the parameters that are typically estimated from statistical distributions that are commonly used in reliability engineering.

Table A.5.3-2: Parameters Typically Estimated from Statistical Distributions

Distribution	True Parameter	Estimated Parameter
Poisson	Occurrence Rate, λ	Sample Occurrence Rate: $\hat{\lambda} = \mathbf{n}/\mathbf{t}$ n = number of observed failures t = period (time, length, volume) over which failures are observed
Binomial	Proportion, **p**	Sample Proportion: $\hat{\mathbf{p}} = \mathbf{x}/\mathbf{n}$ x = number of "successful" trials n = number of statistically independent sample units
Exponential	Mean, θ	Sample Mean: $\hat{\theta} = \bar{\mathbf{x}} = \dfrac{\sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{x}_{\mathbf{i}}}{\mathbf{n}}$ x_i = individual times to failure for each of the observations of sample size "n" n = number of statistically independent sample observations
Normal	Mean, μ	Sample Mean: $\bar{\mathbf{x}} = \dfrac{\sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{x}_{\mathbf{i}}}{\mathbf{n}}$ x_i = individual times to failure for each of the observations of sample size "n" n = number of statistically independent sample observations
	Variance, σ^2	Sample Variance: $\mathbf{s}^2 = \dfrac{\sum_{\mathbf{i}=1}^{\mathbf{n}} (\mathbf{x}_{\mathbf{i}} - \bar{\mathbf{x}})^2}{\mathbf{n}-1}$ s^2 = sample variance (standard deviation, s, equals $(s^2)^{0.5}$) x_i = individual measurements for each of the observations of sample size "n" n = number of statistically independent sample observations

Table A.5.3-2: Parameters Typically Estimated from Statistical Distributions (continued)

Distribution	True Parameter	Estimated Parameter
Weibull	Shape Parameter, β	The estimate of the Weibull shape parameter is: $\hat{\beta} = \frac{1.283}{\mathbf{s}}$ where, $\mathbf{s} = \left(\frac{\sum_{\mathbf{i}=1}^{\mathbf{n}} (\mathbf{x_i} - \bar{\mathbf{x}})^2}{\mathbf{n} - 1} \right)^{0.5}$ $\bar{\mathbf{x}} = \frac{\sum_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{x_i}}{\mathbf{n}}$ s = sample standard deviation; x_i = individual times to failure for each observation of sample size "n"; n = number of statistically independent sample observations
	Scale Parameter, α	The estimate of the Weibull scale parameter is: $\hat{\alpha} = \exp(\bar{\mathbf{x}} + (0.5772)(0.7797)\mathbf{s})$ s = sample standard deviation; x_i = individual measurements for each observation of sample size "n"; n = number of statistically independent sample observations

The parameter estimates shown in Table A.5.3-2 are rather simplistic and easy to use. There are more rigorous techniques available that do a better, more accurate job of estimating parameters, but their complexity in manual use and in definition requires a greater understanding of statistics and mathematical theory than is intended to be covered in this Handbook. Suffice it to say that the references provided at the end of this section provide the additional insight into the mathematics required to understand these techniques. There are also many commercially available statistical data packages that automate these techniques of parameter estimation. Even general-use programs such as Microsoft Excel have basic data analysis tools that can perform parameter estimation. Therefore, it is not necessary to do more within this section than provide a basic definition of what these techniques are.

Table A.5.3-3 includes a very brief discussion of the following parameter estimation techniques:

- Maximum Likelihood Estimation (MLE)
- Least Squares
- Method of Moments
- Bayesian

Table A.5.3-3: Techniques for Parameter Estimation

Technique	Discussion	Process
Maximum Likelihood Estimation (MLE)	In all practical cases, MLE's converge stochastically to the population value. If a MLE exists uniquely and a sufficient statistic for the parameter exists, the MLE is a function of the sufficient statistic. Sometimes the MLE is impossible to find in closed form, and numerical methods must be used (typical of time-domain software reliability models). MLE's are the best estimators for large sample sizes.	1. Express the joint probability density function of the random variables of interest as a function of the unknown parameters (i.e., the likelihood function) 2. Where appropriate, take the natural logarithm of the likelihood function 3. Differentiate the likelihood (or log likelihood) function with respect to each parameter 4. Set all derivatives equal to zero and solve for the parameters as functions of realizations of the random variables 5. Check second-order conditions
Least Squares	Least square estimators may be better when small or medium sample sizes are involved, since they may have smaller bias, or approach normality faster. Least squares estimation minimizes the variance around the estimated parameter. The technique is familiar to those comfortable with linear regression modeling.	1. Express the sum of the squared distance between actual and predicted values as a function of parameter estimates 2. Determine the parameter estimators that minimize the sum of this squared distance (typically using differential calculus)
Method of Moments	This technique works by equating statistical sample moments calculated from a data set to actual population moments. Population moments are determined by the parameters to be estimated. As many moments are equated as there are parameters to be estimated. In most cases of practical interest, these can be found in closed form., but their theoretical justification is not as rigorous as for other parameter estimation methods.	1. Determine the distribution whose parameters are to be estimated (suppose there are "n" parameters to be estimated) 2. Find the first "n" moments of the distribution, either around zero, or around the mean for moments higher than the first 3. Equate these moments to sample moments 4. Solve for the parameters as a function of the realizations of the random variables in the sample.

Table A.5.3-3: Techniques for Parameter Estimation (continued)

Technique	Discussion	Process
Bayesian	Provides an efficient method for incorporating various subjective and objective data sources into parameter estimation. It is a much less practical method than MLE, as the analysis is much more complex and the computation is much more complicated. The validity of the approach is dependent on validity of the model and prior distributions.	1. Assign a non-informative or subjective distribution to the parameters of the model (the "priors"). The priors express the uncertainties in the parameter values. 2. Combine actual data with the "priors" to obtain new parameter distributions (the "posteriors"). The posteriors provide estimates and Bayesian confidence limits for the parameters, producing more precise estimates.

For More Information:

1. Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008
2. Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987, ISBN 007044093X
3. Musa, J.D., "Software Reliability Engineering: More Reliable Software, Faster Development and Testing", McGraw-Hill, July 1998, ISBN 0079132715
4. Nelson, W., "Applied Life Data Analysis", John Wiley & Sons, 1982, ISBN0471094587
5. http://www.math.uah.edu/stat/point/index.xhtml

Appendix A.5.4: Confidence Bounds

Since point estimates are constructed from data that exhibits random variation, these estimates will not be exactly equal to the unknown population parameters. Confidence bounds provide a convention for making statements about the random variation in the estimates of parameters.

Table A.5.4-1: Confidence Bounds for the Poisson Distribution

<table>
<tr><th>Parameter</th><th>One-Sided Confidence Interval</th><th>Two-Sided Confidence Interval</th></tr>
<tr><td colspan="3">Given: The estimate for a the true occurrence rate, λ, is the sample occurrence rate:
$$\hat{\lambda} = \mathbf{n}/\mathbf{t}$$
where,
n = number of observed failures
t = period (time, length, volume) over which failures are observed</td></tr>
<tr><td rowspan="4">True Occurrence Rate, λ</td><td colspan="2">Poisson Limits (approximate only):
Exact confidence levels cannot be conveniently obtained for discrete distributions</td></tr>
<tr><td>$\lambda_L = 0.5\chi^2\left[1-\gamma; 2\mathbf{n}\right]/\mathbf{t}$
$\lambda_U = 0.5\chi^2\left[\gamma; (2\mathbf{n}+2)\right]/\mathbf{t}$</td><td>$\lambda_L = 0.5\chi^2\left[(1-\gamma)/2; 2\mathbf{n}\right]/\mathbf{t}$
$\lambda_U = 0.5\chi^2\left[(1+\gamma)/2; (2\mathbf{n}+2)\right]/\mathbf{t}$</td></tr>
<tr><td colspan="2">Normal Approximation
When “n” is large (say, >10)</td></tr>
<tr><td>$\lambda_L \cong \hat{\lambda} - \mathbf{z}_\gamma (\hat{\lambda}/\mathbf{t})^{0.5}$
$\lambda_U \cong \hat{\lambda} + \mathbf{z}_\gamma (\hat{\lambda}/\mathbf{t})^{0.5}$</td><td>$\lambda_L \cong \hat{\lambda} - \mathbf{z}_{(1+\gamma)/2} (\hat{\lambda}/\mathbf{t})^{0.5}$
$\lambda_U \cong \hat{\lambda} + \mathbf{z}_{(1+\gamma)/2} (\hat{\lambda}/\mathbf{t})^{0.5}$</td></tr>
<tr><td colspan="3">Given: Given the observed rate of occurrence above, the prediction for the future rate of occurrence is:
$$\hat{\mathbf{y}} = \hat{\lambda}\mathbf{s} = (\mathbf{n}/\mathbf{t})\mathbf{s}$$
where,
n, t = as defined above
s = period (time, length, volume) over which future observation is predicted</td></tr>
<tr><td rowspan="4">Future Occurrence Rate, y</td><td colspan="2">Poisson Limits (approximate only)
Closest integer solutions for y_L and y_U from the following equations</td></tr>
<tr><td>$\frac{\mathbf{y}_U}{\mathbf{s}} = \frac{(\mathbf{n}+1)}{\mathbf{t}}\mathbf{F}\left[\gamma; (2\mathbf{n}+2); 2\mathbf{y}_U\right]$
$\frac{\mathbf{s}}{(\mathbf{y}_L+1)} = \frac{\mathbf{t}}{\mathbf{n}}\mathbf{F}\left[\gamma; (2\mathbf{y}_L+2); 2\mathbf{n}\right]$</td><td>$\frac{\mathbf{y}_U}{\mathbf{s}} = \frac{(\mathbf{n}+1)}{\mathbf{t}}\mathbf{F}\left[(1+\gamma)/2; (2\mathbf{n}+2); 2\mathbf{y}_U\right]$
$\frac{\mathbf{s}}{(\mathbf{y}_L+1)} = \frac{\mathbf{t}}{\mathbf{n}}\mathbf{F}\left[(1+\gamma)/2; (2\mathbf{y}_L+2); 2\mathbf{n}\right]$</td></tr>
<tr><td colspan="2">Normal Approximation
When “n” and “y” are large (e.g., each is > 10)</td></tr>
<tr><td>$\mathbf{y}_L \cong \hat{\mathbf{y}} - \mathbf{z}_\gamma \left(\hat{\lambda}\mathbf{s}(\mathbf{t}+\mathbf{s})/\mathbf{t}\right)^{0.5}$
$\mathbf{y}_U \cong \hat{\mathbf{y}} + \mathbf{z}_\gamma \left(\hat{\lambda}\mathbf{s}(\mathbf{t}+\mathbf{s})/\mathbf{t}\right)^{0.5}$</td><td>$\mathbf{y}_L \cong \hat{\mathbf{y}} - \mathbf{z}_{(1+\gamma)/2} \left(\hat{\lambda}\mathbf{s}(\mathbf{t}+\mathbf{s})/\mathbf{t}\right)^{0.5}$
$\mathbf{y}_U \cong \hat{\mathbf{y}} + \mathbf{z}_{(1+\gamma)/2} \left(\hat{\lambda}\mathbf{s}(\mathbf{t}+\mathbf{s})/\mathbf{t}\right)^{0.5}$</td></tr>
</table>

Table A.5.4-2: Confidence Bounds for the Binomial Distribution

<table>
<tr><th>Parameter</th><th>One-Sided Confidence Interval</th><th>Two-Sided Confidence Interval</th></tr>
<tr><td colspan="3">Given: The estimate of the true population proportion, p, is the sample proportion:
$$\hat{p} = x/n$$
where,
x = number of "successful" trials
n = number of statistically independent sample units</td></tr>
<tr><td rowspan="6">True Proportion, p</td><td colspan="2">Binomial Limits (approximate only):
Exact confidence levels cannot be conveniently obtained for discrete distributions</td></tr>
<tr><td>$p_L = \frac{1}{1+(n-x+1)(1/x)F[\gamma;(2n-2x+2);2x]}$
$p_U = \frac{1}{1+(n-x)(1/((x+1)F[\gamma;(2x+2);2n-2x]}$</td><td>$p_L = \frac{1}{1+(n-x+1)(1/x)F[(1+\gamma)/2;(2n-2x+2);2x]}$
$p_U = \frac{1}{1+(n-x)(1/((x+1)F[(1+\gamma)/2;(2x+2);2n-2x]}$</td></tr>
<tr><td colspan="2">Normal Approximation
When "x" and "n-x" are large (e.g., each is > 10)</td></tr>
<tr><td>$p_L \cong \hat{p} - z_\gamma(\hat{p}(1-\hat{p})/n)^{0.5}$
$p_U \cong \hat{p} + z_\gamma(\hat{p}(1-\hat{p})/n)^{0.5}$</td><td>$p_L \cong \hat{p} - z_{(1+\gamma)/2}(\hat{p}(1-\hat{p})/n)^{0.5}$
$p_U \cong \hat{p} + z_{(1+\gamma)/2}(\hat{p}(1-\hat{p})/n)^{0.5}$</td></tr>
<tr><td colspan="2">Poisson Approximation
When "n" is large and "x" is small (e.g., when "x" < n/10)</td></tr>
<tr><td>$p_L \cong 0.5\chi^2[(1-\gamma);2x]/n$
$p_U \cong 0.5\chi^2[\gamma;2x+2]/n$</td><td>$p_L \cong 0.5\chi^2[(1-\gamma)/2;2x]/n$
$p_U \cong 0.5\chi^2[(1+\gamma)/2;2x+2]/n$</td></tr>
<tr><td colspan="3">Given: Given the observed probability above, the prediction for the number of "y" future category units is:
$$\hat{y} = m\hat{p} = m(x/n)$$
where,
x, n = as defined above
m = future sample size</td></tr>
<tr><td rowspan="4">Prediction of Future Probability of "Success", y</td><td colspan="2">Normal Approximation
When "x", "n-x", "y" and "m-y" are all large (say, > 10)</td></tr>
<tr><td>$y_L \cong \hat{y} - z_\gamma[m\hat{p}(1-\hat{p})(m+n)/n]^{0.5}$
$y_U \cong \hat{y} + z_\gamma[m\hat{p}(1-\hat{p})(m+n)/n]^{0.5}$</td><td>$y_L \cong \hat{y} - z_{(1+\gamma)/2}[m\hat{p}(1-\hat{p})(m+n)/n]^{0.5}$
$y_U \cong \hat{y} + z_{(1+\gamma)/2}[m\hat{p}(1-\hat{p})(m+n)/n]^{0.5}$</td></tr>
<tr><td colspan="2">Poisson Approximation
When "n" is large and "x" is small (e.g., when "x" < n/10)
Closest integer solutions for y_L and y_U from the following equations</td></tr>
<tr><td>$\frac{y_U}{m} = \frac{(x+1)}{n}F[\gamma;2x+2;2y_U]$
$\frac{m}{(y_L+1)} = \frac{n}{x}F[\gamma;(2y_L+2);2x]$</td><td>$\frac{y_U}{m} = \frac{(x+1)}{n}F[(1+\gamma)/2;(2x+2);2y_U]$
$\frac{m}{(y_L+1)} = \frac{n}{x}F[(1+\gamma)/2;(2y_L+2);2x]$</td></tr>
</table>

Table A.5.4-3: Confidence Bounds for the Exponential Distribution

Parameter	One-Sided Confidence Interval	Two-Sided Confidence Interval
Given: The estimate of the true population mean, θ, is the sample mean: $\hat{\theta} = \bar{x} = \dfrac{\sum_{i=1}^{n} x_i}{n}$ **where,** **x_i = individual times to failure for each of the observations of sample size "n"** **n = number of statistically independent sample observations**		
	Exponential Limits (exact) for Failure Truncated Tests	
	$\theta_L = \dfrac{2n\bar{x}}{\chi^2[\gamma;2n]}$ $\theta_U = \dfrac{2n\bar{x}}{\chi^2[(1-\gamma);2n]}$	$\theta_L = \dfrac{2n\bar{x}}{\chi^2[(1+\gamma)/2;2n]}$ $\theta_U = \dfrac{2n\bar{x}}{\chi^2[(1-\gamma)/2;2n]}$
True value of the mean, θ	Exponential Limits (exact) for Time Truncated Tests	
	$\theta_L = \dfrac{2n\bar{x}}{\chi^2[\gamma;2(n+1)]}$ $\theta_U = \dfrac{2n\bar{x}}{\chi^2[(1-\gamma);2(n+1)]}$	$\theta_L = \dfrac{2n\bar{x}}{\chi^2[(1+\gamma)/2;2(n+1)]}$ $\theta_U = \dfrac{2n\bar{x}}{\chi^2[(1-\gamma)/2;2n]}$
	Normal Approximation for Failure Truncated Tests When "n" is large (say, > 15)	
	$\theta_L \cong \dfrac{\bar{x}}{\exp\left(z_\gamma / \sqrt{n}\right)}$ $\theta_U \cong \bar{x} * \exp\left(z_\gamma / \sqrt{n}\right)$	$\theta_L \cong \dfrac{\bar{x}}{\exp\left(z_{(1+\gamma)/2} / \sqrt{n}\right)}$ $\theta_U \cong \bar{x} * \exp\left(z_{(1+\gamma)/2} / \sqrt{n}\right)$
Given: The estimate of the true population failure rate, λ, is the sample failure rate: $\hat{\lambda} = \dfrac{1}{\hat{\theta}} = \dfrac{1}{\dfrac{\sum_{i=1}^{n} x_i}{n}}$ **where,** **θ_{hat} = sample mean** **x_i = individual times to failure for each of the observations of sample size "n"** **n = number of statistically independent sample observations**		
	Exponential Limits (exact) for Failure Truncated Tests	
True value of the of the failure rate, λ	$\lambda_L = \dfrac{1}{\theta_U} = \dfrac{\chi^2[(1-\gamma);2n]}{2n\bar{x}}$ $\lambda_U = \dfrac{1}{\theta_L} = \dfrac{\chi^2[\gamma;2n]}{2n\bar{x}}$	$\lambda_L = \dfrac{1}{\theta_U} = \dfrac{\chi^2[(1-\gamma)/2;2n]}{2n\bar{x}}$ $\lambda_U = \dfrac{1}{\theta_L} = \dfrac{\chi^2[(1+\gamma)/2;2n]}{2n\bar{x}}$

Table A.5.4-3: Confidence Bounds for the Exponential Distribution (continued)

<table>
<tr><th>Parameter</th><th>One-Sided Confidence Interval</th><th>Two-Sided Confidence Interval</th></tr>
<tr><td colspan="3">Given: The usual estimate of the 100 pth percentile, y_p, is calculated as:
$$y_p = -\bar{x} * \ln(1-p)$$
where,
p = probability at the 100 pth percentile</td></tr>
<tr><td>True value of the 100 pth percentile, y_p</td><td>$y_{p,L} = -\theta_L * \ln(1-p) = \frac{-2n\bar{x} * \ln(1-p)}{\chi^2[\gamma;2n]}$
$y_{p,U} = -\theta_U * \ln(1-p) = \frac{-2n\bar{x} * \ln(1-p)}{\chi^2[(1-\gamma);2n]}$</td><td>$y_{p,L} = -\theta_L * \ln(1-p) = \frac{-2n\bar{x} * \ln(1-p)}{\chi^2[(1+\gamma)/2;2n]}$
$y_{p,U} = -\theta_U * \ln(1-p) = \frac{-2n\bar{x} * \ln(1-p)}{\chi^2[(1-\gamma)/2;2n]}$</td></tr>
<tr><td colspan="3">Given: The usual estimate of the reliability, R(t), at any age, t, is:
$$R^*(t) = e^{-(t/\bar{x})}$$
where,
R = reliability as a function of time, distance, etc.
t = period at which reliability is assessed (time, distance, etc.)</td></tr>
<tr><td>True value of reliability at end of period, R(t)</td><td>$R_L(t) = e^{-(t/\theta_L)} = \exp\left(-t * \left\{\chi^2[\gamma;2n]/2n\bar{x}\right\}\right)$
$R_U(t) = e^{-(t/\theta_U)} = \exp\left(-t * \left\{\chi^2[(1-\gamma);2n]/2n\bar{x}\right\}\right)$</td><td>$R_L(t) = e^{-(t/\theta_L)} = \exp\left(-t * \left\{\chi^2[(1+\gamma)/2;2n]/2n\bar{x}\right\}\right)$
$R_U(t) = e^{-(t/\theta_U)} = \exp\left(-t * \left\{\chi^2[(1-\gamma)/2;2n]/2n\bar{x}\right\}\right)$</td></tr>
</table>

Table A.5.4-4: Confidence Bounds for the Normal Distribution

<table>
<tr><th>Parameter</th><th>One-Sided Confidence Interval</th><th>Two-Sided Confidence Interval</th></tr>
<tr><td colspan="3">Given: The estimate of the true population mean, μ, is the sample mean:
$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$
where,
x_i = individual times to failure for each of the observations of sample size "n"
n = number of statistically independent sample observations</td></tr>
<tr><td rowspan="2">True value of the mean, μ</td><td colspan="2">Normal Limits (exact)
Also serve as approximate intervals for the mean of a distribution that is not normal</td></tr>
<tr><td>$\mu_L = \bar{x} - t[\gamma;n-1] * \left(s/\sqrt{n}\right)$
$\mu_U = \bar{x} + t[\gamma;n-1] * \left(s/\sqrt{n}\right)$</td><td>$\mu_L = \bar{x} - t[(1-\gamma)/2;n-1] * \left(s/\sqrt{n}\right)$
$\mu_U = \bar{x} + t[(1-\gamma)/2;n-1] * \left(s/\sqrt{n}\right)$</td></tr>
<tr><td colspan="3">Given: The estimate of the true population variance, σ^2, is the sample variance:
$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$
where,
s^2 = sample variance (standard deviation, s, equals $(s^2)^{0.5}$)
x_i = individual measurements for each of the observations of sample size "n"
n = number of statistically independent sample observations</td></tr>
</table>

Table A.5.4-4: Confidence Bounds for the Normal Distribution (continued)

<table>
<tr><th>Parameter</th><th>One-Sided Confidence Interval</th><th>Two-Sided Confidence Interval</th></tr>
<tr><td rowspan="2">True value of the of the variance, σ^2</td><td colspan="2">Normal Limits (exact)</td></tr>
<tr><td>$\sigma_L = s*\left\{\frac{n-1}{\chi^2[\gamma;n-1]}\right\}^{0.5}$
$\sigma_U = s*\left\{\frac{n-1}{\chi^2[(1-\gamma);n-1]}\right\}^{0.5}$</td><td>$\sigma_L = s*\left\{\frac{n-1}{\chi^2[(1+\gamma)/2;n-1]}\right\}^{0.5}$
$\sigma_U = s*\left\{\frac{n-1}{\chi^2[(1-\gamma)/2;n-1]}\right\}^{0.5}$</td></tr>
<tr><td colspan="3">Given: The estimate of the reliability at any age "t", R(t), is:
$$R^*(t) = 1 - \Phi(z)$$
where,
R = reliability as a function of time, distance, etc.
t = period at which reliability is assessed (time, distance, etc.)
Φ(z) = estimate of the fraction of a population failing by age "t"</td></tr>
<tr><td>True value of reliability at end of period, R(t)</td><td>$R_L(t) = 1 - F_U(t) = 1 - \Phi(z_U)$
where
$z = \frac{(x - \bar{x})}{s}$
$z_U \cong z + \frac{z_\gamma}{\sqrt{n}}\left(1 + \frac{z^2(n/2)}{n-1}\right)^{0.5}$

$R_U(t) = 1 - F_L(t) = 1 - \Phi(z_L)$
where
$z = \frac{(x - \bar{x})}{s}$
$z_L \cong z - \frac{z_\gamma}{\sqrt{n}}\left(1 + \frac{z^2(n/2)}{n-1}\right)^{0.5}$</td><td>$R_L(t) = 1 - F_U(t) = 1 - \Phi(z_U)$
where
$z = \frac{(x - \bar{x})}{s}$
$z_U \cong z + \frac{z_{(1+\gamma)/2}}{\sqrt{n}}\left(1 + \frac{z^2(n/2)}{n-1}\right)^{0.5}$

$R_U(t) = 1 - F_L(t) = 1 - \Phi(z_L)$
where
$z = \frac{(x - \bar{x})}{s}$
$z_L \cong z - \frac{z_{(1+\gamma)/2}}{\sqrt{n}}\left(1 + \frac{z^2(n/2)}{n-1}\right)^{0.5}$</td></tr>
</table>

Table A.5.4-5: Confidence Bounds for the Weibull Distribution

<table>
<tr><th>Parameter</th><th>One-Sided Confidence Interval</th><th>Two-Sided Confidence Interval</th></tr>
<tr><td colspan="3">Given: The estimate of the Weibull shape parameter, β, is given as:

$$\hat{\beta} = \frac{1.283}{\mathbf{s}}$$

where,

$$\mathbf{s} = \left(\frac{\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})^2}{\mathbf{n} - 1} \right)^{0.5}$$

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{n} \mathbf{x}_i}{\mathbf{n}}$$

where,
s = sample standard deviation
x_i = individual times to failure for each observation of sample size “n”
n = number of statistically independent sample observations</td></tr>
<tr><td rowspan="2">True value of the Weibull shape parameter, β</td><td colspan="2">Weibull Limits (approximate)
Limits are crude unless “n” is quite large (say, “n” > 100)</td></tr>
<tr><td>$$\beta_L \cong \frac{1}{0.7797\mathbf{s} * \exp\left(\frac{1.049\mathbf{z}_\gamma}{\sqrt{\mathbf{n}}}\right)}$$ $$\beta_U \cong \frac{0.7797\mathbf{s}}{\exp\left(\frac{1.049\mathbf{z}_\gamma}{\sqrt{\mathbf{n}}}\right)}$$</td><td>$$\beta_L \cong \frac{1}{0.7797\mathbf{s} * \exp\left(\frac{1.049\mathbf{z}_{(1+\gamma)/2}}{\sqrt{\mathbf{n}}}\right)}$$ $$\beta_U \cong \frac{0.7797\mathbf{s}}{\exp\left(\frac{1.049\mathbf{z}_{(1+\gamma)/2}}{\sqrt{\mathbf{n}}}\right)}$$</td></tr>
<tr><td colspan="3">Given: The estimate of the Weibull scale parameter, α, is:

$$\hat{\alpha} = \exp\left(\bar{\mathbf{x}} + (0.5772)(0.7797)\mathbf{s}\right)$$

where,
s = sample standard deviation
x_i = individual measurements for each observation of sample size “n”
n = number of statistically independent sample observations</td></tr>
<tr><td rowspan="2">True value of the Weibull scale parameter, α</td><td colspan="2">Weibull Limits (approximate)
Limits are crude unless “n” is quite large (say, “n” > 100)</td></tr>
<tr><td>$$\alpha_L \cong \exp\left((\bar{\mathbf{x}} + 0.45\mathbf{s}) - \mathbf{z}_\gamma \frac{(1.081)(0.7797)\mathbf{s}}{\sqrt{\mathbf{n}}} \right)$$ $$\alpha_U \cong \exp\left((\bar{\mathbf{x}} + 0.45\mathbf{s}) + \mathbf{z}_\alpha \frac{(1.081)(0.7797)\mathbf{s}}{\sqrt{\mathbf{n}}} \right)$$</td><td>$$\alpha_L \cong \exp\left((\bar{\mathbf{x}} + 0.45\mathbf{s}) - \mathbf{z}_{(1+\gamma)/2} \frac{(1.081)(0.7797)\mathbf{s}}{\sqrt{\mathbf{n}}} \right)$$ $$\alpha_U \cong \exp\left((\bar{\mathbf{x}} + 0.45\mathbf{s}) + \mathbf{z}_{(1+\gamma)/2} \frac{(1.081)(0.7797)\mathbf{s}}{\sqrt{\mathbf{n}}} \right)$$</td></tr>
</table>

Table A.5.4-5: Confidence Bounds for the Weibull Distribution (continued)

Parameter	One-Sided Confidence Interval	Two-Sided Confidence Interval
Given: The estimate of the reliability at any age "t", R(t), is: $$\mathbf{R}^*(\mathbf{t}) = \mathbf{e}^{-\left(\frac{t}{\alpha}\right)^{\beta}}$$ **where,** **R = reliability as a function of time, distance, etc.** **t = period at which reliability is assessed (time, distance, etc.)** **α = Weibull scale parameter** **β = Weibull shape parameter** Note: In the source that contained the original Weibull confidence limits for R(t), the value of "t" was expressed as "ln(t)". We believe that this is an error. We have mathematically justified that the correct form of the upper and lower confidence limits for R(t) is as displayed below.		
True value of reliability at end of period, R(t)	Limits are crude unless "n" is quite large (say, "n" > 100) One-sided approximate Weibull limits: $$R_L(t) = \exp\left[-\exp\left(\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)+z_\gamma\left[\frac{1.168+(1.1)\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)^2-(0.1913)\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)}{n}\right]^{0.5}\right)\right]$$ $$R_U(t) = \exp\left[-\exp\left(\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)-z_\gamma\left[\frac{1.168+(1.1)\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)^2-(0.1913)\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)}{n}\right]^{0.5}\right)\right]$$ Two-sided approximate Weibull limits: $$R_L(t) = \exp\left[-\exp\left(\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)+z_{(1-\gamma)/2}\left[\frac{1.168+(1.1)\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)^2-(0.1913)\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)}{n}\right]^{0.5}\right)\right]$$ $$R_U(t) = \exp\left[-\exp\left(\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)-z_{(1-\gamma)/2}\left[\frac{1.168+(1.1)\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)^2-(0.1913)\left(\frac{t-(\bar{x}+0.45s)}{0.7797s}\right)}{n}\right]^{0.5}\right)\right]$$	

Appendix B: Software Reliability Resources

Reliability Education Sources

The following is a compilation of sources for various types of reliability training that include software reliability. This should in no way be considered a complete listing. For further information on any item, contact the cited source directly.

Academic Courses in Software Reliability

University of Maryland
http://www.enre.umd.edu/centers.htm
Center for Risk and Reliability
Glenn L. Martin Hall (088)
Room 0151
College Park, MD 20742-2115
Phone: 301 405-5226

North Carolina State University
http://www.csc.ncsu.edu/
Department of Computer Science
890 Oval Drive, Box 8206
Engineering Building II
Raleigh, NC 27695-8206
Phone: 919-515-2858
Fax: 919-515-7896

Carnegie Mellon University
http://www.cs.cmu.edu/
Electrical and Computer Engineering (ECE)
5000 Forbes Avenue
Pittsburgh, PA 15213-3890
Phone: 412-268-7400
Fax: 412-268-2860

Colorado State University
http://www.cs.colostate.edu/cstop/index.html
Department of Computer Science
279 Computer Science Building
1100 Center Avenue
Fort Collins, CO 80523
Phone: (970) 491-5792
FAX**:**(970) 491-2466

Software Reliability Short Courses

Reliability Information Analysis Center
http://theRIAC.org
6000 Flanagan Rd.
Suite 3
Utica, NY 13502-1348
Phone: 877-363-RIAC (7422) or 315-351-4200
Fax: 315-351-4209

Ops A La Carte
http://www.opsalacarte.com/Pages/education/edu_23swreliability.htm
990 Richard Ave., Suite 101
Santa Clara, CA 95050
Phone: **408-654-0499**
Fax: 408-986-8154

SoHaR Incorporated
http://www.sohar.com
5731 W Slauson Ave., Suite 175
Culver City, CA 90230
Phone: 1-310-338-0990
Fax: 1-310-338-0999

SoftRel
http://www.softrel.com
Phone: 321-514-4659
Fax: 321-821-1948

IEEE Reliability Society Tutorial Videos
http://rs.ieee.org/education.html

Many other sources offer individual engineering courses or individual short courses on reliability engineering topics.

Software Reliability-Related Periodicals

IEEE Transactions on Reliability (Quarterly)

http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=24

IEEE
PO Box 1331
Piscataway, NJ 08855-1331
Phone: 908-981-0060

Journal of the Reliability Information Analysis Center (Quarterly)

http://theRIAC.org

Reliability Information Analysis Center
6000 Flanagan Rd.
Suite 3
Utica, NY 13502-1348
Phone: 877-363-RIAC (7422) or 315-351-4200
Fax: 315-351-4209

Software Testing, Verification and Reliability Journal (Quarterly)

http://www.wiley.com/WileyCDA/WileyTitle/productCd-STVR.html

Wiley
10475 Crosspoint Blvd.
Indianapolis, IN 46256
Phone: 877-762-2974
Fax: 800-597-3299

IEEE Transactions on Software Engineering

http://www.computer.org/portal/web/tse/

IEEE Computer Society
PO Box 3014
Los Alamitos, CA 90720-1314
Phone: 714-821-8380
Fax: 714-821-9975

The Journal of Cyber Security & Information Systems (Quarterly)

https://www.thecsiac.com/

Cyber Security & Information Systems Information Analysis Center (CSIAC)
100 Seymour Road, Suite C102
Utica, NY 13502-1311
Phone: 800-214-7921

The R&M Engineering Journal - Reliability Review (Monthly)

http://www.asq.org/reliability/

American Society for Quality
611 E. Wisconsin Avenue
Milwaukee, WI 53202
Phone: 800-248-1946

The Journal of Systems and Software (Monthly)

http://www.elsevier.com/wps/find/journaldescription.cws_home/505732/description#description

Elsevier
3251 Riverport Lane
Maryland Heights, MO 63043
Phone: 877-839-7126
Fax: 314-447-8077

Software Reliability-Related Symposia and Workshops

Annual Reliability and Maintainability Symposium

http://www.rams.org

IEEE Conference Services
445 Hoes Lane
PO Box 1331
Piscataway, NJ 08855-1331
Phone: 908-562-3878

International Symposium on Software Reliability Engineering (ISSRE)

http://2012.issre.net/

IEEE Conference Services
445 Hoes Lane
PO Box 1331
Piscataway, NJ 08855-1331
Phone: 908-562-3878

Software Reliability-Related Texts

1. Lyu, M.R., “Handbook of Software Reliability Engineering”, Computer Society Press, ISBN: 0-07-039400-8, 1996
2. Musa, J.D., “Software Reliability Engineering: More Reliable Software, Faster Development and Testing”, McGraw-Hill, ISBN: 0-07-913271-5, 1998
3. Musa, J.D., “Software Reliability Engineering: More Reliable Software Faster and Cheaper – Second Edition”, Authorhouse, ISBN: 1-4184-9387-2, 2004
4. Peled, D.A., “Software Reliability Methods”, Springer-Verlag, ISBN: 0-387-95106-7, 2001
5. Pham, H., “Software Reliability”, Springer-Verlag, ISBN: 981-3083-84-0, 2000
6. Gritzalis, D., “Reliability, Quality and Safety of Software-Intensive Systems”, Chapman and Hall, ISBN: 0-412-80280-5, 1997
7. Jones, C., “Software Assessments, Benchmarks and Best Practices”, Addison Wesley, ISBN: 0-201-48542-7, 2000
8. Neufelder, A.M., “Ensuring Software Reliability”, Marcel Dekker, Inc., ISBN: 0-824-78762-5, 1993
9. Pressman, R.S., “Software Engineering: A Practitioner’s Approach – 5th Edition”, McGraw-Hill, ISBN: 0-073-65578-3, June 2000
10. Fenton, N.E. and Pfleeger, S.L., “Software Metrics: A Rigorous and Practical Approach”, International Thomson Publishing, ISBN: 0-534-95425-1, May 1998
11. Grady, R.B., “Practical Software Metrics for Project Management and Process Improvement”, Prentice-Hall, ISBN: 0-137-20384-5, 1992
12. Musa, J.D., Iannino, A., and Okumoto, K., “Software Reliability: Measurement, Prediction, Application”, McGraw-Hill, ISBN: 0-070-44093-X, May 1987
13. Mili, A., “An Introduction to Program Fault Tolerance – A Structured Programming Approach”, Prentice-Hall, ASIN: 0-134-93551-X, 1990
14. Boehm, B.W., “Software Engineering Economics”, Prentice-Hall, ISBN: 0-138-22122-7, 1981
15. Rook, P., “Software Reliability Handbook”, Elsevier Applied Science, ISBN: 1-851-66400-9, June 1990
16. Gilb, T. and Graham, D., “Software Inspection”, Addison-Wesley, ISBN: 0-201-63181-4, 1993
17. Pressman, R.S., “Software Engineering: A Practitioner’s Approach – 4th Edition”, McGraw-Hill, ISBN: 0-070-52182-4, 1997
18. “Software System Safety Handbook”, Joint software System Safety Committee, December 1999
19. Beizer, B., “Black-Box Testing: Techniques for Functional Testing of Software and Systems”, John Wiley and Sons, ISBN: 0-471-12094-4l May 1995
20. Dunn, R.H., Ullman, R.S., “TQM for Computer Software”, McGraw-Hill, ISBN: 0-070-18314-7, 1994

Software Reliability Related Organizations

- Center for Experimental Software Engineering
- Center for Systems and Software Engineering
- Centre for Software Reliability
- Data and Analysis Center for Software (DACS)
- IBM: Center for Software Engineering
- Reliability Information Analysis Center (RIAC)
- Software Engineering Institute
- Software Technology Support Center

Appendix C: Tools to Support Software Reliability

This Appendix introduces and provides sources for automated tools that perform or help with software reliability analyses and tasks. Why use a tool for software reliability analysis?

- To handle the large amount of data
- To do number crunching
- To facilitate what-if analyses
- To provide structure and organization

The universe of tools available for software reliability is still much smaller than those developed for hardware and physical components of systems. For some types of reliability analyses, a "hardware" or "systems" reliability tool can be relatively easily adapted to automate analyses for which no "software" tool is available.

<u>Benefits of Using Automated Tools</u>

- Allows comparison over time (if normalized), across projects, even with other organizations (against benchmarks)
- Automation increases likelihood of use
- Reduces chance of calculation error
- Results are more easily replicated
- May provide data/outputs to feed into other development/environment tools
- Provides documentation artifacts to facilitate communication with management, customers, and other non-software stakeholders.
- Reduces workload when applying several models simultaneously to determine the best fit for an organization/project/process (as recommended by Brocklehurst and Littlewood in Reference 1), as well as when recalibrating a model to a specific project

<u>Limitations of Automated Tools</u>

- Tools do not provide a complete solution. It is still necessary to define and collect data
- Any tool needs to be calibrated to the environment in which it is used
- The output requires skilled interpretation
- Using a tool will not solve a reliability problem. A misapplied tool or misinterpreted results may even harm a project
- Tools have not been developed for all models or techniques
- Tool interfaces may not be user-friendly or intuitive

<u>Considerations in Selecting Tools</u>:

- Tool selection depends on the tasks to be done, the form of the input data and the form desired for the output of the programs. Additional tools may be required, such as least squares fit programs for handling resources usage data (see Reference 2)

- It may be better to write your own tools for reliability analyses. Those with the skill levels needed to run, understand, and interpret the results of a tool tend to have programming experience, tool

- Consider the availability of tools for the desired analyses. If no tools are commercially available, the software reliability functions will need to include tool development time in the schedule during the project planning stages

- Consider the amount of automatic data collection. To minimize the impact on the project's schedule, automated collection tools should be considered whenever possible. Factors to weigh in deciding to automate data collection include: Is there a commercial off-the-shelf tool available or must it be developed? What is the cost involved in either the purchase of the tool or its development? When will the tool be available? If it must be developed, will its development schedule coincide with the planned use?

- What impact will the data collection process have on the development schedule? Can the tool handle adjustments that may be needed? Can the adjustments be completed in a timely manner? How much overhead (people and time) will be needed to keep the data collection process going? (see Reference 3)

- Flexibility should be designed into the tool, as data collection requirements may change. Consider ways of ensuring the right data are being gathered. Make some type of assessment of not only what the tool saves in time and resources but also how the data collection process is improved

- To determine what to spend on a tool (either purchasing a COTS tool or developing a custom tool), estimate the amount of time and effort that would be expended if the data had been collected or the analyses performed manually. These statistics yield cost estimates that can be compared with the procurement and implementation costs of the automated tool. If the cost of the automated tool is significantly higher, question the wisdom of acquiring or developing the tool. However, even if the costs come out higher, consideration must be given to future uses of the tool (i.e., long-term life cycle cost savings. Once the tool has been developed or acquired it may be easily adapted over many software development efforts and could yield significant savings. (see Reference 3)

- Plan to provide training for all concerned parties in the use of the tool, as well as how it benefits the overall process over the long run (see Reference 3)

A comprehensive set of tools should include the capability (see Reference 2) to (1) compute present failure intensity from failure intervals and calendar time, (2) plot successive results from the first tool (3) perform simulations, i.e., run the first (two) tools with hypothesized data, (4) convert raw failure log data to failure intervals, and (5) perform a least squares fit of data.

The following sections provide information on automated software reliability tools in specific categories:

Appendix C.1: Software Reliability Prediction
Appendix C.2: Software Reliability Estimation
Appendix C.3: Software Reliability Growth
Appendix C.4: Software Metrics
Appendix C.5: Software Test Coverage
Appendix C.6: Miscellaneous Software Reliability
Appendix C.7: System Reliability

Each section includes a table that provides the tool name, a brief description of the tool, and source or contact information. Web addresses are included wherever possible. Finally, Section 9.8 provides a look at tools that are under development at universities and in research labs as examples of what may eventually become commercially available. A summary of the tools identified in subsequent sections is presented in Table C.0-1. The summary shows which types of software reliability analyses each tool supports.

Table C.0-1: Automated Software Reliability Tool Summary

Tool Name	Tool Function(s)						
	Prediction (C.1)	Estimation (C.2)	Growth (C.3)	Metrics (C.4)	Test Coverage (C.5)	Miscellaneous (C.6)	System Reliability (C.7)
217Plus							X
ARM						X	
BlockSim	X						X
CASRE		X					
CA - Test Coverage					X		
DevPartner					X		
ENVY				X			
ESTM		X					
Ferret					X		
FREstimate	X						
Goel-Okumoto		X					
GRASP						X	
McCabe IQ2				X	X		
MEADEP							X
M-elopee						X	
METRIC				X			
PC/UX-Metric				X			
QA C				X			
RAM-ILS							X
Rational Pure Coverage					X		
Reliability & Maintenance Analyst	X						
RG			X				
SARA			X				
SEER-DFM							X
SilkTest					X		
SIMUL8						X	
SLIM	X	X		X			
SLIM-Metrics				X			
SMERFS		X					

Tool Name	Tool Function(s)						
	Prediction (C.1)	Estimation (C.2)	Growth (C.3)	Metrics (C.4)	Test Coverage (C.5)	Miscellaneous (C.6)	System Reliability
SoftRel		X					
SoRel		X	X				
SRE		X					
SRMP		X					
STEER		X					
SW Rel Pred	X		X				
TCA					X		
TestWorks					X		
TestWorks/ Advisor				X			
TFD	X						X
WhenToStop			X				
BullseyeCoverage					X		
Clover					X		
CodeTEST					X		
Coverage Meter					X		
CTC++					X		
Dynamic Code Coverage					X		
GCT					X		
Insure++					X		
Java Test Coverage					X		
JavaCov					X		
Koalog Code Coverage					X		
LDRA Testbed					X		
McCabe IQ					X		
Rational Test RealTime					X		
TCAT C/C++, Java					X		

For More Information:

1. Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008

2. Musa, J.D.; Iannino, A.; and Okumoto, K.; "Software Reliability: Measurement, Prediction, Application", McGraw-Hill, May 1987, ISBN 007044093X

3. "Recommended Practice: Software Reliability", ANSI/AIAA R-013-1992, American Institute of Aeronautics and Astronautics (AIAA), Washington, DC.

4. Rook, P., ed, "Software Reliability Handbook", Center for Software Reliability (CSR), City University of London, Elsevier, Chapman & Hall Ltd, ISBN 1851664009

5. http://www.incose.org/ProductsPubs/products/toolsdatabase.aspx

Appendix C.1: Software Reliability Prediction Tools

Reliability prediction tools are applied in the earlier phases of the software life cycle. They can be tied in with project management and computer-aided software engineering (CASE) tools included in software engineering environments, or be a part of a larger toolset. Table C.1-1 provides a representative sample of what is currently available on the market.

Table C.1-1: Sample Software Reliability Prediction Tools

Tool Name	Description	Source
FREstimate	This software reliability prediction tool is used early in development, as early as the concept phase to predict the delivered or fielded failure rate or MTTF of a software system. The software reliability prediction methods are based on historical data from similar previously fielded software projects in which the actual MTTF, failure rate or reliability is known.	SoftRel Ann Marie (Leone) Neufelder PO Box 588 Sugarland, TX 77487-0588 281-494-5982 http://www.softrel.com/prod01.htm
Reliability & Maintenance Analyst	Reliability analysis software package. The life data analysis module estimates the distribution parameters for Weibull, normal, lognormal, and exponential distributions. Parameters can be estimated using maximum likelihood (MLE), probability plotting, hazard plotting, and moment matching. Features include Bayesian estimation zero-failure test planning, support for the 3-parameter Weibull distribution, complete, singly- and multiply-censored, and grouped data, and for graphical and statistical goodness-of-fit tests for the time to fail and reliability. Computes confidence limits. Also includes a maintenance optimization module.	Engineered Software, Inc. 3710 Briarbrooke Lane Rochester, MI 48306 248-276-2276 http://www.engineeredsoftware.com/rma.asp
SLIM (Software Lifecycle Management)	Consists of four products: SLIM-Estimate, SLIM-Control, SLIM-Metrics, and Estimate Express. Together, they use an organization's own process productivity and staffing metrics to predict software reliability over time and generate metrics for project tracking and control.	Quantitative Software Management Inc. 2000 Corporate Ridge McLean, VA 22102 800-424-6755; 703-790-0055 FAX: 703-749-3795 http://www.qsm.com/
SW Rel Prediction	Predicts fault density based on empirical data relating fault density to the process capability of the underlying development process. Transforms the latent fault density into an exponential reliability growth curve over time.	Sam Keene PO Box 337 Lyons, CO 80540 (303) 684 2277 s.keene@ieee.org

Appendix C.2: Software Reliability Estimation Tools

Reliability estimation tools can be used at several points in the software life cycle. Typically, they are applied once testing begins and failure data is available. Some of these tools are designed to be used throughout a product's operational life as well.

Table C.2-1: Software Reliability Estimation Tools

Tool Name	Description	Source
* AT&T Software Reliability Engineering (SRE) Toolkit	Executes Musa basic and Musa-Okumoto logarithmic Poisson execution time models. Accepts both time domain and interval domain failure data. Estimates total failures, and the initial and present failure rates (failure intensity), and includes confidence intervals.	This toolkit was developed at what is now AT&T Labs. They no longer distribute or support it, but it is supported by: Dr. Laurie Williams Associate Professor North Carolina State University Department of Computer Science 890 Oval Drive, Engineering Building 2, Room 3272 Campus Box 8206 Raleigh, NC 27695-8206 USA Phone: (919)513-4151 Fax: (919)515-7896 williams@csc.ncsu.edu http://collaboration.csc.ncsu.edu/laurie/
Computer-Aided Software Reliability Estimation (CASRE) Tool	Calculates present reliability and predicts future reliability as a function of test time, represented in terms of reliability measures such as cumulative number of failures, failures per time interval, and the product's reliability function. Provides product reliability estimates during system testing and field operation. Allows users to select and apply existing models from the library of the SMERFS tool. Two categories of models are used, depending on the type of input data: time-between-failures models take the sequence of times between failures as the input while failure-count models take number of failures per interval as the input.	This tool was originally developed by NASA's JPL, and until July, 1998 was distributed by COSMIC at the University of Georgia. Distribution is now available through the Open Channel Foundation: http://www.openchannelsoftware.com/projects/CASRE_3.0
Goel-Okumoto Nonhomogeneous Poisson Process Software Reliability Model	Automated version of the model. Finds maximum likelihood estimators of model parameters using Newton-Raphson or Muller's method; does goodness-of-fit tests based on a Kolmogorov-Smirnov statistic; estimates remaining faults, cumulative failures, and reliability; and estimates optimal release time based on certain cost criteria.	Available from the Data & Analysis Center for Software http://www.thedacs.com/about/services/goel.php

Table C.2-1: Software Reliability Estimation Tools (continued)

Tool Name	Description	Source
Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS)	Consists of a driver program and a library of reliability models. Highly flexible: accepts both time and interval domain data, allows users to tailor the interface, add or remove models in the library, and develop custom drivers.	This tool was originally developed at the Naval Surface Warfare Center, but is no longer available from them. It is included on the Data and Tool CD in Reference 1.
***** SoftRel	A software reliability process simulator that captures the effects of interrelationships among activities, and characterizes all events as piecewise-Poisson Markov processes with the defined event rate functions in a software project. Simulates both defects in specification documents and faults in code.	Developed by Robert C. Tausworthe at NASA's Jet Propulsion Laboratory. Included on the Data and Tool CD in Reference 1.
Software Reliability Program (SoRel)	Does reliability growth tests and applies reliability growth models. Allows inter-failure and failure intensity data. Evaluates mean time to next failure, the intensity function, the cumulative number of failures and the residual failure rate. Reliability growth tests are: arithmetical mean, Laplace, Kendall and Spearmann. Reliability growth models are: Goel-Okumoto NHPP; Littlewood-Verrall failure rate; Kanoun-Laprie hyperexponential; and Yamada S-Shaped. Model validation criteria are Kolmogorov-Smirnov distance, prequential likelihood and residue or relative residue.	Karama Kanoun LAAS-CNRS 7 avenue du Colonel Roche 31077 Toulouse Cedex 4 France Tel: 05 61 33 62 00 Fax: 05 61 55 35 77 http://www.laas.fr/surf/sorel/sorel.html
Software Reliability Modeling Programs (SRMP)	Contains nine models, uses maximum likelihood estimation to compute the model parameters, and calculates: reliability function, failure rate, mean time to failure, median time to failure, and the parameters for each model. Runs on time domain input data only. Allows analysis of goodness-of-fit for the models.	Dr. Bev Littlewood Center for Software Reliability City University London London, England Tel. +44 71 477 8420 http://www.csr.city.ac.uk/people/bev.littlewood
STEER	Estimates the number of defects in the software at delivery/start of operation by fitting actual defect discovery data to an assumed equation. Defect data is obtained from the development and testing process, commencing with design inspections.	John Gaffney, Jr. gaffney123@verizon.net
* **(Reference 1)** These tools are available on the CD ROM that comes with the book Lyu, M.R. (Editor), "Handbook of Software Reliability Engineering", McGraw-Hill, April 1996, ISBN 0070394008		

Appendix C.3: Software Reliability Growth Tools

Reliability growth is an unfamiliar concept for most software engineers. Software developers instead tend to see reliability growth as progress in testing, or as part of quality assurance, and that perception is reflected in the relative lack of tools in this list. Some researchers use trend analysis to approximate reliability growth, implying that trend analysis tools could be adapted for software reliability growth studies.

Table C.3-1: Software Reliability Growth Tools

Tool Name	Description	Source
RG	RG is designed for analyzing Reliability Growth data and trends utilizing most growth models, such as NHPP (AMSAA), Duane, Gompertz, Modified Gompertz, Lloyd Lipow and Logistic. This tool is not strictly for software analysis, but its highly configurable interface accommodates software-related input data.	ReliaSoft ReliaSoft Plaza, Suite 103 115 S. Sherwood Village Drive Tucson, AZ, 85710 888-722-7522; 952-953-3292 **Fax:** 520-886-0399 http://www.reliasoft.com/rga/index.htm
Software Assurance Reliability Automation (SARA) Tool	The Software Assurance Reliability Automation Tool (SARA) is a comprehensive system which incorporates both reliability growth modeling and design code metrics for analyzing software time between failure data.	Software Assurance Technology Center NASA Goddard Space Flight Center 8800 Greenbelt Road Greenbelt, MD 20771
WhenToStop	This software reliability tool can be used during testing, once there are observed failures. It can be used to estimate whether or not the required or predicted failure rate or MTTF objective will be met.	SoftRel Ann Marie (Leone) Neufelder PO Box 588 Sugarland, TX 77487-0588 281-494-5982 http://www.softrel.com/prod02.htm

Appendix C.4: Software Metrics Tools

Software metrics is a more common area for commercial tool development and availability. The relationships between measurable characteristics of code (as opposed to artifacts from earlier in the software life cycle) and software engineering management goals are more well-known. A number of commonly-used metrics have been developed, over the last 20+ years or so, of software engineering research and development (McCabe, Halstead, the Rome Lab quality framework). Metrics are easily tracked and reported to management.

Table C.4-1: Software Metrics Tools

Tool Name	Description	Source
ENVY/QA	Provides a system of quality assurance tools for software professionals. Tools include Code Metrics, Code Critic, Code Coverage, Code Publisher and Code Formatter. The Code Metrics tool gathers 38 static metrics on methods, classes, applications and configuration maps. Report sections are customizable. Thresholds can be defined for each metric. Users can view all results or focus on methods outside of the thresholds.	SilverMark, Inc. 9650 Strickland Road, Suite 103 PMB 251 Raleigh, NC 27615-1937 email: info@oti.com http://www.silvermark.com/Product/smalltalk/va/stm/envyQA.html
McCabe IQ	A tightly integrated suite of tools, consisting of: QA, Test, Reengineer, TRUEtrack, TRUEchange, Testcompress. QA computes the essential McCabe Metrics. Test implements basis path testing. The other tools provide additional support for testing, configuration management, and analysis of existing systems.	McCabe & Associates, Inc. 9861 Broken Land Pkwy. Columbia, MD 21046 1-800-638-6316 401-572-3100 http://www.mccabe.com/products.htm
METRIC	Software Metrics Processor/Generator. Computes software metrics for source code, including Halstead software science metrics and cyclomatic complexity metrics. Reports metrics in configurable reports and charts.	Software Research, Inc. 1663 Mission Street San Francisco, CA 94103 USA Phone: +1 (415) 861-2800 FAX: +1 (415) 861-9801 http://www.soft.com/Products/Advisor/metric.html
PC-Metric	Analyzes C, C++, COBOL, FORTRAN, Pascal, Modula-2, BASIC, Ada, and dBase programs' source code and produces metrics to determine complexity. Provides cross-reference feature that lists lines on which each variable is used in each function or procedure.	SET Laboratories Inc. 26976 S. Highway 213 Mulino, OR 97042 503-829-7123 FAX: 503-829-7220 http://www.molalla.net/~setlabs/pcmetric.html
QA C, QA C for PC, QA C++	Analyzes C or C++ code prior to compilation. Provides configurable warning messages. Produces over 45 industry-accepted metrics. Reports on ISO and ANSI C conformance. Produces variety of graphical output reports. Highlights portability problems. Detects language errors. Establishes software quality benchmark.	Programming Research Ltd. Mark House 9/11 Queens Road Hersham Surrey KT12 5LU United Kingdom Tel: +44 (0) 1932 88 80 80 Fax: +44 (0) 1932 88 80 81
SLIM-Metrics	Windows based version of the PADS (Productivity Analysis Database System) measurement and metrics repository. Captures metrics on resources, schedule, reliability, and tool and method information.	Quantitative Software Management Inc. 2000 Corporate Ridge McLean, VA 22102 800-424-6755; 703-790-0055 FAX: 703-749-3795 http://www.qsm.com/slim_metrics.html
TestWorks/ Advisor	Provides static source code analysis and measurement to establish and measure quality benchmarks with metrics, analyze source code for anomalies with static analysis, and automatically generate a wide variety of test data. Includes 17 metrics.	Software Research, Inc. 1663 Mission Street San Francisco, CA 94103 USA Phone: +1 (415) 861-2800 FAX: +1 (415) 861-9801 http://www.soft.com/Products/Advisor/index.html

Appendix C.5: Software Test Coverage Tools

Testing is the category for which software tools are most abundant. Software testing has long been the focus of commercial tool development (and research) because the relationship is so obvious; the causes and effects seem easily quantified. Even the most hapless (CMMI Level 0 - chaotic) software development organization does some testing, and may approach it as a panicked realization that this is the last/only chance to get it right, or at least shippable. Test coverage tools, as a subset of testing tools, help determine the scope of the testing effort for planning, for monitoring its progress, and for determining when enough testing has been done. An up to date list of Test Coverage Tools can be found at http://www.testingfaqs.org/t-eval.html#BullseyeCoverage. Below is a detailed list as of the date of this publication.

Table C.5-1: Software Test Coverage Tools

Tool Name	Language	Source/Info.
BullseyeCoverage	C++/C on Microsoft and Unix operating systems	BullseyeCoverage http://www.bullseye.com/
Clover	Java	Atlassian http://www.atlassian.com/software/clover/
CodeTEST	C/C++ for embedded systems software	FreeScale/CodeWarrior http://www.freescale.com/webapp/sps/site/homepage.jsp?nodeId=012726
CoverageMeter	C, C++, C#	CoverageMeter http://www.coveragemeter.com/
CTC++	C and C++	Testwell http://www.testwell.fi/ctcdesc.html
Dynamic Code Coverage	C and C++	Dynamic Memory Solutions http://www.dynamic-memory.com/
GCT	C test coverage (freeware)	Gct-Request@cs.uiuc.edu
Insure++	C, C++	ParaSoft Corporation http://www.parasoft.com/
Java Test Coverage	Java	Semantic Designs, Inc. http://www.semdesigns.com/Products/TestCoverage/index.html
JavaCov	Java	Alvicom
Koalog Code Coverage	Java	Koalog SARL
LDRA Testbed	C, C++, Ada83, Ada95 & Assemblers (Intel, Freescale and Texas Instruments)	LRDA Software Technology http://www.ldra.com/testbed.asp
McCabe IQ	Ada, ASM86, C, C#, C++.NET, C++, COBOL, FORTRAN, JAVA (Eclipse IDE also available), JSP, Perl, PL1, VB, VB.NET	McCabe Software, Inc. http://www.mccabe.com/iq.htm
Rational Test RealTime	Java, C/C++, Ada	IBM Rational http://www-01.ibm.com/software/rational/
TCAT C/C++, Java	C, C++, Java	Software Research, Inc. http://www.soft.com/TestWorks/

Appendix C.6: Miscellaneous Software Reliability Tools

These tools are not strictly designed for reliability analysis, but can be used to support software reliability-related tasks.

Table C.6-1: Miscellaneous Software Reliability Tools

Tool Name	Description	Source/Info.
Automated Requirement Measurement (ARM) Tool	An early life cycle tool for assessing requirements specified in natural language. The tool provides measures for project managers to assess the quality of a requirements specification document. The tool is not intended to evaluate the correctness of the specified requirements, but is an aid to "writing the requirements right," not "writing the right requirements."	Software Assurance Technology Center NASA Goddard Space Flight Center 8800 Greenbelt Road Greenbelt, MD 20771
GRASP	Creates Control Structure Diagrams (CSD), an algorithmic level graphical representation for software control flow and data structure designed to fit in the space normally taken by indentation in source code. CSD improves the comprehension efficiency of source code and, therefore increases reliability and reduces costs.	Dr. James H. Cross II, Chair Dept. of Computer Science & Eng. 107 Dunstan Hall Auburn University, AL 36849 http://www.eng.auburn.edu/department/cse/research/grasp/
M-élopée (Software Assessment from Test through Exploitation)	A CASE tool for software reliability, code quality measurement, statistical testing and software reliability modeling, covering the final phases of the life cycle: testing, validation, and operational use. Provides complete management of reliability data, trend calculation, modeling and simulation and management decision support.	Mathix 19 rue du Banquier 75013 Paris Tel: 01 43 37 76 0 Fax: 01 43 37 00 73 /

Appendix C.7: System Reliability Tools

In performing a complete reliability analysis for a system that contains both software and hardware, the better system reliability tools will necessarily include some facility for handling the reliability of the software components. The caution here is that the tools could be overkill in terms of cost and complexity for an organization that produces only software.

Table C.7-1: Selected System Reliability Tools

Tool Name	Description	Source
BlockSim	Evaluates complex system reliability, availability and maintainability. Performs exact computations and predictions for system reliability analysis and optimization. Systems are defined via a Reliability Block Diagram (RBD) approach, where the blocks are components, subassemblies, assemblies, failure modes with multiple properties, or encapsulations of other blocks.	ReliaSoft ReliaSoft Plaza, Suite 103 115 S. Sherwood Village Drive Tucson, AZ, 85710 888-722-7522 952-953-3292 **Fax:** 520-886-0399 http://www.reliasoft.com/products.htm
Measurement-based Dependability (MEADEP)	System-oriented reliability and availability measurement, modeling and prediction tool. Analysis of degraded-mode operations and recovery scenarios. Supports RBDs, Markov modeling, and MTBF calculations.	SoHar Inc. 8421 Wilshire Blvd., Suite 201 Beverly Hills, CA 90211 323-653-4717 x300 FAX: (323) 653-3624 http://www.sohar.com/software/meadep/
217Plus	Framework for system reliability assessment. Predicts inherent and field MTBF. The 217Plus concept accounts for the myriad of factors that can influence system reliability, combining all those factors into an integrated system reliability assessment resource. 217Plus was developed to overcome limitations in MIL-HDBK-217.	Reliability Information Analysis Center 100 Seymour Rd Suite C 101 Utica, NY 13502-1311 315.351.4200 877.363.RIAC (Toll Free) http://www.theriac.org/
RAM - Design Evaluation Workbench	Includes modules for MTBF analysis, Block Diagram Evaluation (BDE) using both steady-state or Monte Carlo methods, and Fault & Success Tree Analysis (FTA). Program calculations include reliability, availability, sensitivity analysis, and spares deficits.	Management Sciences, Inc. 6022 Constitution Ave, NE Albuquerque NM 87110 505-255-8611 Fax : 505-268-6696 http://www.mgtsciences.com/
SEER-DFM	System Evaluation and Estimation of Resources (SEER).Design for Manufacturability/Assembly tool for determining optimum product design and manufacturing methods and processes. Design for Cost, Design for Manufacturability & Design for Assembly analysis.	Galorath Incorporated 100 North Sepulveda Blvd, Suite 1801 El Segundo, CA 90245 Phone 310-414-3222 Fax 310-414-3220
Tools for Decision (TFD)	TFD software supports decision-making in the disciplines of life cycle cost, optimal stocking of spare parts, level of repair analysis, reliability prediction, and systems modeling. It is applicable from the earliest stages of acquisition decision making (front-end analysis) throughout the acquisition, through-life, or in-service period.	Systems Exchange/TFD Group PO Box 3290 Monterey, CA 93942 831 649 3800 831 649 3866 fax sei@tfdg.com

Appendix D: Acronyms

α	Producer's Risk
β	Consumer's Risk
λ	Failure Rate (1/Mean Time Between Failure)
μ	Arithmetic Mean
μ	Repair Rate (1/Mean Corrective Maintenance Time)
σ	Standard Deviation
$\hat{\theta}$	Observed Point Estimate Mean Time Between Failure
θ_0	Upper Test (Design Goal) Mean Time Between Failure
θ_1	Lower Test (Unacceptable) Mean Time Between Failure
θ_D	Demonstrated MTBF (Controlled Testing)
θ_P	Predicted Mean Time Between Failure

3M	Maintenance, Material, Management System
6σ	Six Sigma Statistical Process Control

A_a	Achieved Availability
A_i	Inherent Availability
A_{IC}	Airborne Inhabited Cargo
A_{IF}	Airborne Inhabited Fighter
A_m	Materiel Availability
A_o	Operational Availability
A_{UC}	Airborne Uninhabited Cargo
A_{UF}	Airborne Uninhabited Fighter
AAA	Allocations Assessment and Analysis
ACAT	Acquisition Category
ACC	Air Combat Command
ACO	Administrative Contracting Officer
ACPM	AMSAA Crow Projection Model
ACQ	Acquisition
ADAS	Architecture Design and Assessment System
ADM	Advanced Development Model
ADP	Automatic Data Processing
ADPE	Automatic Data Processing Equipment
ADT	Administrative Delay Time
AETC	Air Education and Training Command
AETG	
AFAE	Air Force Acquisition Executive
AFFSA	Air Force Flight Standards Agency
AFIT	Air Force Institute of Technology
AFLMA	Air Force Logistics Management Agency
AFMC	Air Force Materiel Command
AFOTEC	Air Force Operational Test and Evaluation Center
AFR	Air Force Regulation
AFSOC	Air Force Special Operations Command
AFSPC	Air Force Space Command
AFTO	Air Force Technical Order
AGS	Ambiguity Group Size
AI	Artificial Intelligence
ALC	Air Logistics Center
ALT	Accelerated Life Test
ALU	Arithmetic Logic Unit
AMC	Air Mobility Command
AMEC	Army Management Engineering College
AMGS	Automatic Microcode Generation System
AMPM	AMSAA Maturity Projection Model
AMSAA	Army Materiel Systems Analysis Activity
AMSDL	Acquisition Management Systems and Data Control List
ANOVA	Analysis of Variance
ANSI	American National Standards Institute
AoA	Analysis of Alternatives
AOTR	Assessment of Operational Test Readiness
APB	Acquisition Program Baseline
APTE	Automatic Programmed Test Equipment
APUC	Average Unit Procurement Cost
AR	Adjusted Rank
ARM	Anti-radiation Missile
ARP	Armament Recording Program
A_{RW}	Airborne Rotary Wing
ASA	Advanced Systems Architecture
ASC	Aeronautical Systems Center
ASQC	American Society of Quality Control
ASR	Acquisition Strategy
ASR	Alternative System Review
ASTM	American Society for Testing and Materials
AT&L	Acquisition, Technology and Logistics
ATC	Air Training Command
ATE	Automatic/Automated Test Equipment
ATF	Advanced Tactical Fighter
ATG	Automatic Test Generation
ATP	Acceptance Test Procedure
ATPG	Automatic Test Pattern Generator
ATTD	Advanced Technology Transition Demonstration
AVIP	Avionics Integrity Program

b	Billion
b	BIT
bps, B/S	Bits Per Second
BAFO	Best and Final Offer
BCC	Block Check-Sum Character
BCS	Bench Check Serviceable
BCWP	Budget Cost of Work Performed
BCWS	Budget Cost of Work Scheduled
BEA	Budget Estimate Agreement
BELL	Bell Labs
BES	Budget Estimate Submission
BFT	Between Failure Arrival Time
BIST	Built-in Self Test
BIT	Built-In-Test
BITE	Built-In-Test Equipment
BIU	Bus Interface Unit
BLER	Block Error Rate
BLRIP	Beyond Low-Rate Initial Production
BMD	Ballistic Missile Defense
BPPBS	Biennial Planning, Programming, and Budgeting System

C	Centigrade
C_p	Process Capability Index
C_{pk}	Process Performance Index
C^3	Command, Control and Communications
C^3CM	Command, Control, Communications and Countermeasures
C^3I	Command, Control, Communications Intelligence
CA	Contracting Activity
CA	Corrective Action
CAD	Computer Aided Design
CADBIT	Computer Aided Design for Built-In Test
CAE	Computer Aided Engineering
CAE	Component Acquisition Executive
CAIG	Cost Analysis Improvement Group
CALS	Computer Aided Acquisition Logistics & Support
CAM	Computer-Aided Manufacturing
CAP	Corrective Action Period
CARD	Cost Analysis Requirements Document
CAS	Column Address Strobe
CAS	Computer Aided Support
CASE	Computer-Aided Software Engineering
CASS	Computer Aided Schematic System
CAT	Computer Aided Test
CBA	Capabilities Based Assessment
CCB	Capacitive Coupled BIT
CCB	Configuration Control Board
CDD	Capability Development Document
CDF	Cumulative Density Function
CDR	Critical Design Review

CDRL	Contract Data Requirements List
CE	Concurrent Engineering
CEO	Chief Executive Officer
CEST	Software Cost Estimation
CFAR	Constant False Alarm Rate
CFE	Contractor Furnished Equipment
CFSR	Contract Fund Status Report
CI	Configuration Item
CIM	Computer Integrated Manufacturing
CINC	Commander-In-Chief
CISC	Complex Instruction Set Computer
CIU	Control Interface Unit
CLIN	Contract Line Item Number
CLS	Client Server Technology
cm	Centimeter
CM	Configuration Manager or Management
CM	Corrective Maintenance
CML	Current Mode Logic
CMM	Capability Maturity Model
CMMI	
CND	Can Not Duplicate
CNI	Communications, Navigation and Identification
CO	Contracting Officer
CODEC	Coder Decoder
COI	Critical Operational Issue
COIC	Critical Operational Issue and Criteria
COMM	Communications
COMSEC	Communications Security
CONOPS	Concept of Operations
COPS	Complex Operations Per Second
COTS	Commercial Off-The-Shelf
CPCI	Computer Program Configuration Item
CPD	Capability Production Document
CPFF	Cost-Plus-Fixed-Fee
CPIF	Cost-Plus-Incentive-Fee
CPM	Control Processor Module
CPSC	Consumer Product Safety Commission
CPU	Central Processing Unit
CR	Clean Room
CRC	Cyclic Redundancy Check
CRTA	Critical Reliability Technology Assessment
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSP	Common Signal Processor
CSR	Control Status Register
CSU	Computer Software Unit

df	Degrees of Freedom
df_{err}	Degrees of Freedom for the Error
df_F	Degrees of Freedom for a Factor
D-Level	Depot Level
DAB	Defense Acquisition Board
DACS	Data and Analysis Center for Software
DAG	Defense Acquisition Guidebook
DAMIR	Defense Acquisition Management Information Retrieval
DC	Duty Cycle
DCAPE	Director of Cost Assessment and Program Evaluation
DDR&E	Director of Defense, Research and Engineering
DDT&E	Director of Development Test and Evaluation
DECTED	Double Error Correcting, Triple Error Detecting
DED	Double Error Detection
DEM/VAL	Demonstration and Validation
DESC	Defense Electronics Supply Center
DFARS	Defense Federal Acquisition Regulation Supplement
DFMEA	Design Failure Mode and Effects Analysis
DFR	Design for Reliability
DHS	Department of Homeland Security
DID	Data Item Description
DIP	Dual In-Line Package
DISC	Defense Industrial Supply Center
DLA	Defense Logistics Agency
DMR	Defense Management Review
DoD	Department of Defense
DoD-ADL	Department of Defense Authorized Data List
DOE	Design of Experiments
DOS	Disk Operating System
DOT&E	Director, Operational Test and Evaluation
DOTMLPF	Doctrine, Training, Materiel, Leadership, Personnel and Facilities
DP	Data Processor
DP	Development Planning
DR	Design Review
DR	Discrimination Ratio
DRS	Deficiency Reporting System
DSE	Director of Systems Engineering
DSP	Digital Signal Processing
DT	Development Test
DT&E	Development Test and Evaluation
DT/OT	Development Test/Operational Test
DTIC	Defense Technical Information Center
DUT	Device Under Test

EC	Electronic Commerce
ECC	Error Checking and Correction
ECCM	Electronic Counter Countermeasures
ECF	Effective Cumulative Failures
ECM	Electronic Countermeasures
ECP	Engineering Change Proposal
ECS	Environmental Control System
ECU	Environmental Control Unit
EDA	Electronic Design Automation
EDAC	Error Detection and Correction
EDM	Engineering Development Model
EEC	European Economic Community
EGS	Electronic Ground System
EGSE	Electronic Ground Support Equipment
EIA	Electronics Industries Association
EMD	Engineering and Manufacturing Development
EoA	Evaluation of Alternatives
EPA	Environmental Protection Agency
ER	Established Reliability
ESC	Electronic System Center
ESD	Event Sequence Diagrams
ESM	Electronics Support Measure
ESS	Environmental Stress Screening
ET	Event Tree
ETE	Electronic or External Test Equipment
ETT	Expected Test Time
EUT	Early User Test
EVA	Extreme Value Analysis
EW	Electronic Warfare
EXP	Exponent

ftp	File Transfer Protocol
F	F-Ratio Statistic
F/W	Firmware
FA	False Alarm
FAA	Federal Aviation Administration
FAR	False Alarm Rate
FAR	Federal Acquisition Regulation
FARR	Forward Area Alerting Radar Receiver
FAT	First Article Testing
FBT	Functional Board Test
FCA	Functional Configuration Audit
FD	Fault Detection
FD/SC	Failure Definition and Scoring Criteria
FDI	Fault Detection and Isolation
FEF	Fix Effectiveness Factor
FES	First Engine Shutdown
FFD	Fraction of Faults Detected
FFI	Fraction of Faults Isolated
FFP	Firm Fixed Price
FFRP	Field Failure Return Program
FFT	Fast Fourier Transform
FFTAU	Fast Fourier Transform Arithmetic Unit
FFTCU	Fast Fourier Transform Control Unit
FH	Flight Hours
FI	Fault Isolation
FIFO	First In – First Out
FIO	Failure Intensity Objective
FIR	Fault Isolation Resolution
FIRO	Failure Intensity Reduction Objective
FIT	Fault Isolation Test
FITS	Failures Per 10^9 hours
FLIR	Forward Looking Infrared
FLOPS	Floating Point Operations Per Second

FMC Full Mission Capability
FMEA Failure Modes and Effects Analysis
FMECA Failure Modes, Effects and Criticality Analysis
FOC Full Operational Capability
FOM Figure of Merit
FOV Field of View
FP Floating Point; Function Point
FPMFH Failures Per Million Flight Hours
FPMH Failures Per Million Hours
FQR Formal Qualification Review
FQT Final Qualification Test
FR Failure Rate
FRACAS Failure Reporting and Corrective Action System
FRB Failure Review Board
FRP Full Rate Production
FS Full Scale
FSA Functional Solution Assessment
FSD Full Scale Development
FSED Full Scale Engineering Development
FT Fault Tree
FTA Fault Tree Analysis
FTF Fault Tolerance Fraction
FY Fiscal Year

G_B Ground Benign
G_F Ground Fixed
G_M Ground Mobile
GAO General Accounting Office
GD Global Defect
GEIA Government Electronics & Information Technology Association
GFE Government Furnished Equipment
GFP Government Furnished Property
GIDEP Government Industry Data Exchange Program
GIMADS Generic Integrated Maintenance Diagnostic
GM Global Memory
GOCO Government Owned Contractor Operated
GOMAC Government Microcircuit Applications Conference
GOTS Government Off-the-Shelf
GSE Ground Support Equipment
GSPA Generic Signal Processor Architecture
GUI Graphical User Interface

html HyperText Markup Language
http HyperText Transmission Protocol
HALT Highly Accelerated Life Test
HASS Highly Accelerated Stress Screening
HAST Highly Accelerated Stress Test
HDBK Handbook
HDL Hardware Description Language
HDS Hierarchical Design System
HHDL Hierarchical Hardware Description Language
HOL Higher Order Language
HOQ House of Quality
HPP Homogeneous Poisson Process

I-Level Intermediate Level
I/O Input/Output
IAC Information Analysis Center
IAW In Accordance With
IBR Integrated Baseline Review
ICD Interface Control Document
ICD Initial Capabilities Document
ICNIA Integrated Communications, Navigation and Identification Avionics
ICWG Interface Control Working Group
ICE Independent cost Estimates
ID Integrated Diagnostics
IDAS Integrated Design Automation System
IDHS Intelligence Data Handling System
IEC International Electrotechnical Commission
IEEE Institute of Electrical and Electronic Engineers
IEST Institute of Environmental Science and Technology
IETM Interactive Electronic Technical Manuals
IF Interface
IFB Invitation for Bid
IFF Identification Friend or Foe
IG Inspector General
ILA Integrated Logistics Assessment
ILS Integrated Logistics Support
ILSM Integrated Logistics Support Manager
INEWS Integrated Electronic Warfare System
IOC Initial Operational Capability
IOT&E Initial Operational Test & Evaluation
IPD Integrated Product Development
IR Inverted Rank
IR&D Independent Research & Development
ISA Instruction Set Architecture
ISR In-Service Review
ISO International Standards Organization
ISPS Instruction Set Processor Specification
IT Information Technology
ITAR International Traffic in Arms Regulation
ITM Integrated Test and Maintenance
ITR Initial Technical Review
IV&V Independent Verification and Validation
IWSM Integrated Weapons System Management

JAN Joint Army Navy
JCIDS Joint Capabilities Integration and Development System
JCS Joint Chiefs of Staff
JROC Joint Requirements Oversight Council
JSC Johnson Space Center
JTAG Joint Test Action Group

k Boltzmann's Constant (8.65 x 10^{-5} electron volts/°Kelvin
K Kelvin
K Thousand
KHB Kennedy Handbook
KM/DS Knowledge Management/Decision Support
KOPS Thousands of Operations per Second
KPP Key Performance Parameter
KSA Key System Attribute

LAN Local Area Network
LCB Lower Confidence Bound
LCC Life Cycle Cost
LCL Lower Confidence Limit
LCS Life Cycle Sustainment
LCSP Life Cycle Sustainment Plan
LDT Logistic Delay Time
LEX Life Extension
LFR Launch and Flight Reliability
LHR Low Hop Rate
LIFO Last In First Out
LISP List Processing
LOC Lines of Code
LRIP Low Rate Initial Production
LRM Line Replaceable Module
LRU Line Replaceable Unit
LSA Logistics Support Analysis
LSAR Logistics Support Analysis Record
LSB Least Significant Bit
LSE Lead System Engineer
LSI Large Scale Integration
LSL Lower Specification Limit
LSSD Level Sensitive Scan Design
LUT Look Up Table
LUT Limited User Test

ms Millisecond
M Maintainability
M Million
M_{ct} Mean Corrective Maintenance Time
Mhz Megahertz
M-Demo Maintainability Demonstration
M-MM Mean Maintenance Manhours
M&S Modeling and Simulation
MAIS Major Automated Information Systems
MAJCOM Major Command
MAP Modular Avionics Package

MB	Megabyte
MBPS	Million Bits Per Second
MCA	Monte Carlo Analysis
MCBF	Mean Cycles Between Failure
MCCR	Mission Critical Computer Resources
MCFOS	Military Computer Family Operating System
MCOPS	Million Complex Operations Per Second
MCTL	Military Critical Technology List
MCU	Microcontrol Unit
MDA	Milestone Decision Authority
MDAP	Major Defense Acquisition Program
MDCS	Maintenance Data Collection System
MDD	Material Development Decision
MDT	Mean Downtime
MDT	Maintenance Downtime
MENS	Mission Element Needs Statement
MENS	Mission Equipment Needs Statement
MESL	Mission-Essential Systems (or Subsystems) List
MFHBF	Mean Flying Hours Between Failure
MFHBMCF	Mean Flying Hours Between Mission Critical Failures
MFHBUMA	Mean Flying Hours Between Unscheduled Maintenance Actions
MFLOPS	Million Floating Point Operations Per Second
MIL	Military
MIL-STD	Military Standard
MIN	Maintenance Interface Network
MIPS	Million Instructions Per Second
MISD	Multiple Instructions Single Data
MLD	Master Logic Diagram
MLE	Maximum Likelihood Estimation
MLH/OH	
MLIPS	Million Logic Inferences/Instructions Per Second
MMBF	Mean Miles Between Failure
MMD	Mean Mission Duration
MMH/FH	Maintenance Manhours Per Flight Hour
MMH/PH	Maintenance Manhours Per Possessed Hour
MMPS	Million Multiples Per Second
MMR	Multimode Radar
MN	Maintenance Node
MNN	Maintenance Network Node
MNS	Mission Need Statement
MOA	Memorandum of Agreement
MOE	Measure of Effectiveness
MOP	Measure of Performance
MOPS	Million Operations Per Second
MOTS	
MP	Maintenance Processor
MPCAG	Military Parts Control Advisory Group
MPMT	Mean Preventive Maintenance Time
MR	Maintenance Ratio
MR	Median Rank
MRBF	Mean Rounds Between Failure
MS	Management Strategy
MS A	Milestone A
MS B	Milestone B
MS C	Milestone C
MSB	Most Significant Bit
MSE	Mean Square Error
MST	Mean Square of Treatments
MTBCF	Mean-Time-Between-Critical- Failure
MTBD	Mean-Time-Between-Demand
MTBDE	Mean-Time-Between-Downing- Events
MTBF	Mean-Time-Between-Failure
$MTBF_F$	Mean-Time-Between-Failure (Field)
MTBFF	Mean-Time-Between-Functional-Failure
MTBM	Mean Time Between Maintenance
MTBM-IN	Mean-Time-Between- Maintenance-Induced (Type 2 Failure)
MTBM-INH	Mean-Time-Between-Maintenance-Inherent (Type 1 Failure)
MTBM-ND	Mean-Time-Between- Maintenance-No Defect (Type 6 Failure)
MTBM-P	Mean-Time-Between- Maintenance-Preventive
MTBM-TOT	Mean-Time-Between- Maintenance-Total
MTBMA	Mean-Time-Between- Maintenance-Actions
$MTBM_F$	Mean-Time-Between- Maintenance (Field)
$MTBM_S$	Mean Time Between Maintenance-Scheduled
$MTBM_U$	Mean Time Between Maintenance-Unscheduled
MTBR	Mean-Time-Between- Removals
$MTBR_F$	Mean-Time-Between- Removals (Field)
MTBUMA	Mean-Time-Between- Unscheduled-Maintenance- Actions
MTE	Minimal-Test-Equipment
MTE	Multipurpose Test Equipment
MTI	Moving Target Indicator
MTTE	Mean-Time-To-Error
MTTF	Mean-Time-To-Failure
MTTR	Mean-Time-To-Repair
MTTRS	Mean-Time-To-Restore- System
MVP	
MWPS	Million Words Per Second
MWS	Major Weapon Systems

NASA	National Aeronautics and Space Administration
NAVAIR	Naval Air Systems Command
NCSA	National Center for Supercomputing Applications
NDI	Nondevelopmental Item
NDT	Nondestructive Testing
NHB	NASA Handbook
NHPP	Nonhomogeneous Poisson Process
NIST	National Institute of Standards and Technology
ns	Nanosecond
N_S	Naval Sheltered
N_U	Naval Unsheltered
NWSC	Naval Warfare Surface Center

O&M	Operation and Maintenance
O&S	Operation and Support
O-Level	Organizational Level
OC	Ownership Cost
OCI	Organizational Conflicts of Interest
ODC	Orthogonal Defect Classification
OEM	Original Equipment Manufacturer
OMB	Office of Management and Budget
OMS/MP	Operational Mode Summary and Mission Profile
OOD	Object Oriented Design
OPEVAL	Operational Evaluation
OPR	Office of Primary Responsibility
OPS	Operations Per Second
OPTEMPO	Operating Tempo
OPTEVFOR	Operational Test and Evaluation Force
ORD	Operational Requirements Document
OSD	Office of the Secretary of Defense
OSS	Open Source Software
OT	Operational Test
OT&E	Operational Test and Evaluation
OTA	Operational Test Activity
OTRR	Operational Test Readiness Review
OTS	Off-The-Shelf
OUSD(AT&L)	Office of the Undersecretary of Defense for Acquisition, Technology and Logistics
OUSD(PA&E)	Office of the Under Secretary of Defense for Program Analysis and Evaluation

p	Probability
P	Percentile
PACAF	Pacific Air Forces
PAL	Programmable Array Logic
PARCA	Performance Assessments and Root Cause Analysis
PAT	Process Action Team
PAT	Programmable Alarm Thresholds
PAUC	Program Acquisition Unit Cost
PBA	Performance-Based Agreement
PC	Personal Computer
PCA	Physical Configuration Audit
PCO	Procuring Contracting Officer
PDF	Probability Density Function
PDL	Program Design Language
PDR	Preliminary Design Review
PEM	Program Element Monitor
PFMEA	Process Failure Mode and Effects Analysis
PM	Preventive Maintenance
PM	Program Manager
PM2	AMSAA Projection Methodology
PMD	Program Management Directive
PMP	Program Management Plan
PMR	Program Management Review
PMRT	Program Management Responsibility Transfer
PO	Program Office
POL	Petroleum, Oil and Lubricants

PPM	Parts Per Million
PR	Parameter Ratio
PRA	Probabilistic Risk Assessment
PRAT	Production Reliability Acceptance Test
PROTO	Rapid Prototyping
PRR	Production Readiness Review
PRST	Probability Ratio Sequential Test
QA	Quality Assurance
QC	Quality Control
QDR	Quality Deficiency Report
QFD	Quality Function Deployment
QML	Qualified Manufacturers List
QPL	Qualified Parts List
QRAT	Quick Reliability Assessment Tool
QT&E	Qualification Test & Evaluation
QUMR	Quality Unsatisfactory Material Report
R	Reliability
R&D	Research and Development
R&M	Reliability & Maintainability
RADC	Rome Air Development Center
RAM-C	Reliability, Availability, Maintainability and Cost
RAM	Reliability, Availability and Maintainability
RAMS	Reliability and Maintainability Symposium
RCM	Reliability Centered Maintenance
RD	Random Defect
RDGD	Reliability Development Growth Test
RDT	Reliability Demonstration Test
RDT&E	Research, Development, Test and Evaluation
REG	Register
REMIS	Reliability and Maintainability Information System
RFP	Request for Proposal
RGT	Reliability Growth Test
RGTMC	Reliability Growth Tracking Model - Continuous
RIAC	Reliability Information Analysis Center
RISA	Reduced Instruction Set Architecture
RISC	Reduced Instruction Set Computer
RIW	Reliability Improvement Warranty
RL	Rome Laboratory
R_M	Materiel Reliability
RMS	Reliability, Maintainability and Supportability
RMS	Reliability, Maintainability and Safety
RMS	Root Mean Square
ROC	Required Operational Capability
ROS	Reduced Operation Software
ROM	Rough Order of Magnitude
RQT	Reliability Qualification Test
RR	Requirements Review
RSA	Rapid Simulation Aids
RSR	Runtime Status Register
RSS	Root-Sum-Squared
RTL	Register Transfer Language
RTOK	Retest OK
RTQC	Real Time Quality Control
S	Second
S_F	Space Flight
S/N	Serial Number
S/W	Software
SA	Sneak Analysis
SAE	Society of Automotive Engineers
SAF	Secretary of the Air Force
SAI	Statistical Applications Institute
SAR	Synthetic Aperture Radar
SBIR	Small Business Innovative Research
SC	Space Center
SCA	Sneak Circuit Analysis
SDI	Strategic Defense Initiative
SDL	System Descriptive Language
SDLC	Software Development Life Cycle
SDR	System Design Review
SDS	Structured Design System
SE	Simultaneous Engineering
SE	Support Equipment
SE	Systems Engineering
SEC	Software Engineering Criteria
SECDED	Single Error Correction, Double Error Detection
SECDEF	Secretary of Defense
SED	Single Error Detection
SEDS	System Engineering Detailed Schedule
SEMP	Systems Engineering Management Plan
SEP	Systems Engineering Plan
SER	Soft Error Rate
SERD	Support Equipment Recommended Data
SEU	Single Event Upset
SFMEA	Software Failure Mode and Effects Analysis
SFR	System Functional Review
SFT	System Failure Time
SLIM	Software Lifecycle Management
SMD	Standard Military Drawing
SPI	Software Process Improvement
SPL	Software Product Lines
SOA	Safe Operating Area
SOAR	State-of-the-Art Report
SOLE	Society of Logistics Engineers
SON	Statement of Need
SORD	Systems Operational Requirements Document
SOW	Statement of Work
SPC	Statistical Process Control
SPEC	Specification
SPO	System Program Office
SQC	Statistical Quality Control
SRA	Shop Replaceable Assembly
SRD	System Requirement Document
SRR	Systems Requirement Review
SRU	Shop Replaceable Unit
SS	Sum of Squares
SSA	Source Selection Authority
SSAC	Source Selection Advisory Council
SSE	Sum of Squares of Deviations
SSEB	Source Selections Evaluation Board
SSP	Source Selection Plan
SSR	Software Specification Review
SST	Sum of Squares Between Two Tests
ST	Self Test
STD	Standard
STE	Special Test Equipment
STINFO	Scientific and Technical Information
SUT	Statistical Usage Testing
SVR	System Verification Review
t	Time
T&E	Test and Evaluation
T&M	Time and Materials
TAAF	Test, Analyze and Fix
TAC	Tactical Air Command
TBCF	Time Between Critical Failures
TBD	To Be Determined
TBF	Time Between (Successive) Failures
TBM	Time Between Maintenance
TBR	Time Between Removals
TD	Technology Development
TDM	Time Division Multiplexing
TDS	Technology Development Decision
TDS	Technology Development Strategy
TEMP	Test & Evaluation Master Plan
TES	Test and Evaluation Strategy
TET	Technical Evaluation Team
TFOM	Testability Figure of Merit
TLCC	Total Life Cycle Cost
TM	Technical Manuals
TM	Test Modules
TMDE	Test Measurement and Diagnostic Equipment
TMP	Test and Maintenance Processor
TO	Technical Orders
TOC	Total Ownership Cost
TPM	Technical Performance Measure
TPS	Test Program Set
TPWG	Test Plan Working Group
TQM	Total Quality Management
TR	Technical Report
TRA	Technology Readiness Review
TRD	Test Requirements Document
TRL	Technology Readiness Level
TRR	Test Readiness Review

TTSF Time to System Failure

UCL Upper Confidence Limit
URL Uniform Resource Locator
USAF United States Air Force
USAFE United States Air Forces in Europe
USC United States Code
USD(AT&L) Under Secretary of Defense for Acquisition, Technology & Logistics
USL Upper Specification Limit
USN United States Navy
UUT Unit Under Test

V & V Verification and Validation
VAMOSC Visibility and Management of Operating and Support Costs
VOC Voice of the Customer
VSP Variable Site Parameters

WBS Work Breakdown Structure
WFL Waterfall Development Model
WOLF Work Order Logistics File
WRA Weapons Replaceable Assembly
WRSK War Readiness Spares Kit
WSARA Weapon Systems Acquisition Reform Act of 2009
WUC Work Unit Code
WWW World Wide Web

XCVR Transceiver