Contents

1. RELIABILITY MATHEMATICS 1

1.1. A Brief Introduction to Probability Theory and Statistics 1
1.1.1. Basic Definitions 1
1.1.2. Probability Properties, Theorems and Axioms 2
1.1.2.1. Mutually Exclusive Events 2
1.1.2.2. Conditional Probability 2
1.1.2.3. Independent Events 2
1.1.3. Random Variable 3

1.2. The Probability and Cumulative Density (Distribution) Functions 3
1.2.1. Designations 3
1.2.2. Definition 4
1.2.3. Mathematical Relationship Between the pdf and cdf 5
1.2.4. Mean Life (MTTF) 5
1.2.5. Median Life 5
1.2.6. Modal Life 6

1.3. Statistical Distributions used in Reliability Analysis 6
1.3.1. Exponential Distribution 8
1.3.2. The Normal Distribution 10
1.3.3. The Log-Normal Distribution 13
1.3.4. The Weibull Distribution 15

2. BURN IN 21

2.1. Introduction 21
2.1.1. Burn-In Definitions 21
2.1.2. The Differences between Burn-In and Environmental Stress Screening (ESS) 22

2.2. Burn-In Methods and Their Effectiveness 22
2.2.1. Static Burn-In 22
2.2.2. Dynamic Burn-in 23
2.2.3. Test During Burn-In 23
2.2.4. High-Voltage Stress Tests 24

2.3. Burn-In Documents 26

2.4. Burn-In Test Conditions Specified By MIL-STD-883C 26

2.5. Test Temperature 29

2.6. Reliability after Burn-In 30
2.6.1. Residual MTTF 32

2.7. A Physics of Failure Approach to IC Burn-In 32
2.7.1. Burn-In Philosophy 33
2.7.2. Problem with Present Approach to Burn-In 33
2.7.3. A Physics- Of – Failure Approach to Burn –In 38
2.7.3.1. Understanding Steady –State Temperature Effects 38
2.7.3.2. Setting up the Burn-in Profile 39

3. RELIABILITY EVALUATION USING DATABOOKS 43

3.1. MIL-HDBK-217 vs. HALT/HASS 43
3.1.1. The Purpose of MIL-HDBK-217 43
3.1.2. The Merit of HALT/HASS 44
3.1.3. Why MIL-HDBK-217 Turns Out Inaccurate Predictions 46
3.1.4. Conclusion 47
3.1.5. References 48

3.2. A New System-Reliability Assessment Methodology 49
3.2.1. Abstract 49
3.2.2. Background 50
3.2.2.1. Need for the Model 50
3.2.2.2. Uses for Reliability Predictions 51
3.2.2.3. Methodologies Used In Performing Predictions 51
3.2.3. The Basis for a New Model 53
3.2.3.1. Uncertainty in Traditional Approach Estimates 53
3.2.3.2. System Failure Causes 54
3.2.3.3. Model Description 55
3.2.3.4. Initial Assessment 58
3.2.3.5. Process Grading 58
3.2.3.6. Adding Software Failure Rate 59
3.2.3.7. Adding Failure Rate Due to Wearout Modes 62
3.2.3.8. Logistic Failure Rate Contributions 62
3.2.3.9. Adding Empirical Data 63
3.2.4. Future Plans 64
3.2.5. References 64

4. RELIABILITY DESIGN IMPROVEMENT METHODS 67

4.1. Introduction 67

4.2. Derating 67
4.2.1. Importance of Derating 67
4.2.2. Effect of Derating On Part Stress Reliability Prediction 68
4.2.3. Method of Derating 68

4.3. Redundancy 69
4.3.1. Active Parallel Redundancy 69
4.3.2. Standby Redundancy 70
4.3.3. K-out-of-M Redundancy 70

4.4. Stress Reduction 71
4.4.1.1. Reliability Growth Testing 72
4.4.2. Duane Model 72

4.5. Cumulative MTBF 74
4.5.1. Alternate Duane Plot 74
4.5.2. Limitations 74

5. COST ANALYSIS 77

5.1. Life Cycle Cost Analysis 77
5.1.1. The Economics of Reliability and Maintainability and System Design 79
5.1.2. Life-Cycle Cost Model 81

5.2. Warranty Cost Analysis 84

6. ACCELERATED LIFE TESTING DATA ANALYSIS 87

6.1. Introduction 87

6.2. Data and Data Types 89
6.2.1. Complete Data 89
6.2.2. Censored Data 89
6.2.2.1. Censored Type I Data 90
6.2.2.2. Censored Type II Data 90
6.2.2.3. Multi-censored Data 91

6.3. Stress Types and Stress Levels 91

6.4. Life-Stress relationships 92

6.5. Analyzing Data from Accelerated Life Tests 93

6.6. How do you fit an acceleration model? 94
6.6.1. Graphical Method 95
6.6.1.1. Life Distribution Parameters at Each Stress Level 95
6.6.1.2. Life Distribution Probability Plotting 95
6.6.1.3. Determining the X and Y Position of the Plot Points 98
6.6.1.4. Median Ranks 98
6.6.1.5. Some Shortfalls of Manual Probability Plotting 99
6.6.1.6. Life-Stress Relationship Plotting 99
6.6.1.7. How to fit an Arrhenius Model with Graphical Estimation 103
6.6.1.8. Comments on the Graphical Method 105
6.6.2. MLE (Maximum Likelihood) Parameter Estimation 105
6.6.2.1. Background Theory 106
6.6.2.2. Illustrating the MLE Method Using the Exponential Distribution 107
6.6.2.3. Illustrating the MLE Method Using the Normal Distribution 108
6.6.2.4. Estimator 109
6.6.2.5. Unbiased Estimator 109
6.6.3. Conclusions 109

6.7. Calculated Results and Plots 110
6.7.1. Examples of Reporting for Parametric Data Analysis 112
6.7.1.1. Probability Plot 112
6.7.1.2. Reliability Function 112
6.7.1.3. Probability Density Function 113
6.7.1.4. Failure Rate Function 114
6.7.1.5. Life vs. Stress Plot 114
6.7.1.6. Reliability Growth 115

6.8. Confidence Bounds 115
6.8.1. One-Sided and Two-Sided Confidence Bounds 116
6.8.1.1. Two-Sided Bounds 116
6.8.1.2. One-Sided Bounds 116
6.8.1.3. Electronic Devices Example 117

7. HIGHLY ACCELERATED TESTING 127

7.1. Introduction 127

7.2. Why Things Fail? 127
7.2.1. The Bathtub Curve 128

7.3. The Purposes of HALT and HASS 129

7.4. Equipments Required **132**

7.5. Some General Comments on HALT and HASS **134**

8. ACCELERATED LIFE TESTING CONCEPTS AND MODELS 137

8.1. Test Purpose **138**
8.1.1. On Materials 139
8.1.2. On Products 141

8.2. Types of Acceleration and Stress Loading **144**
8.2.1. Overstress Testing 146
8.2.1.1. About Degradation Mechanisms 146
8.2.1.2. Stresses and Stress Levels 147
8.2.1.3. Stress Loading 148

8.3. Types of Accelerated Test Data **152**

8.4. Analysis Method **157**
8.4.1. Life-Stress Models 158
8.4.2. Statistics Based Models 160
8.4.2.1. Exponential Distribution Acceleration Model 160
8.4.2.2. Weibull Distribution Acceleration Model 161
8.4.3. Physics Statistics Based Models 162
8.4.3.1. The Arrhenius Model 162
8.4.3.2. The Eyring Model 163
8.4.3.3. The Inverse Power Rule Model 165
8.4.3.4. Combination Model 166
8.4.4. Physics Experimental Based Models 167
8.4.4.1. Electromigration Model 167
8.4.4.2. Humidity Dependence Failures 168
8.4.4.3. Temperature-Humidity Relationship 169
8.4.4.4. Fatigue Failures 170
8.4.5. Degradation Models 171
8.4.5.1. Resistor Degradation Model 172

9. REPAIRABLE SYSTEM ANALYSIS 175

9.1. Availability and Maintainability Measures **175**
9.1.1. Contributions to unavailability 176

9.2. Availability **176**

9.3. RS Models and Availability **179**
9.3.1. Renewal models 179
9.3.1.1. System Structure and Assumptions 179
9.3.1.2. General Results 180
9.3.1.3. Special Case 180
9.3.1.4. System Availability 181
9.3.2. Minimal Repair Models 182
9.3.2.1. System Structure and Assumptions 182
9.3.2.2. General Results 183
9.3.3. CTMC Models 184
9.3.3.1. Single Machine Problems 184
9.3.3.2. Multiple Machine Problems 185

9.4. Maintainability **188**
9.4.1. Maintainability Impact on Availability 190

9.4.2. Maintainability Measures 190
9.4.2.1. Probability of Task Completion(PTC) 190
9.4.2.2. Mean Duration of Maintenance Task (MDMT) 191
9.4.2.3. Percentage Duration of Maintenance Task(DMT_p) 191
9.4.2.4. Variability of Duration of Maintenance Task (CV(DMT)) 191
9.4.2.5. Success of Task Completion (STC) 192
9.4.2.6. Maintenance Personnel Demand per Maintenance Task (MMPD) 192
9.4.3. Item Based Statistics 194
9.4.3.1. Mean Time in Maintenance (MTIM) 194
9.4.3.2. Mean Time to Restore (MTTR) 194
9.4.3.3. Maintenance Hours per Operational Unit (MHOU) 194
9.4.4. System Based Statistics 194
9.4.5. Other Areas of Maintainability Engineering 195

9.5. Maintenance and Optimization 195
9.5.1. Reactive Maintenance 196
9.5.2. Predictive Maintenance 197
9.5.3. Replacement Decision 198
9.5.4. Inspection Decisions (Inspection Models) 203
9.5.4.1. Optimal inspection frequency: Maximization of profit. 203
9.5.4.2. Optimal inspection frequency: Minimization of downtime. 204
9.5.4.3. Optimal inspection interval to maximize the availability of equipment used in emergency conditions 205

10. SOFTWARE RELIABILITY CONCEPTS 207

10.1. Terminologies 207

10.2. Overview of Software Reliability 208
10.2.1. Errors, Faults and Failures 209
10.2.2. Software failure mechanisms 210

10.3. Software Reliability Metrics 211

10.1. Measurements to assess Reliability 212

10.2. Complimentary strategies to achieve Reliability 212
10.2.1. Fault Avoidance 212
10.2.2. Fault Tolerance 212

10.3. Error Categories 213
10.3.1. Design errors 213
10.3.2. Coding Errors 214
10.3.3. Clerical Errors 214
10.3.4. Debugging errors 214
10.3.5. Testing errors 214

10.4. Failure Classification 215

10.5. Data Collection 215
10.5.1. Data collection procedure 215

10.6. Failure Count Data vs. Execution Time Data 219
10.6.1. Failure-Count Data 219
10.6.2. Execution Time Data 220
10.6.3. Transformations between the Two Types of Input 221

10.7. Software Reliability Engineering 221
10.7.1. What It Is and Why It Works 222

10.7.2. A Proven, Standard, Widespread Best Practice 223

10.8. Software Reliability Measurements 226
10.8.1. Software reliability estimation 226
10.8.2. Software reliability prediction 226

10.9. Type of Tests in SRE 226
10.9.1. Reliability growth test 227
10.9.2. Certification test 227

10.10. Software Reliability Engineered Testing 228
10.10.1. Definitions 229
10.10.2. SRET Steps 230

10.11. SRE Process and Fone Follower Example 233
10.11.1. Define the Product 234
10.11.2. Implement Operational Profiles 235
10.11.3. Define "Just Right" Reliability 237
10.11.4. Prepare For Test 238
10.11.5. Execute Test 238
10.11.6. Guide Test 239
10.11.7. Collect Field Data 241

10.12. Conclusion 241

11. SOFTWARE TESTING 243

11.1. Introduction 243

11.2. Key Concepts 246
11.2.1. Correctness Testing 246
11.2.2. Performance testing 249
11.2.3. Reliability testing 250
11.2.4. Security testing 250

11.3. Testing Automation 251

11.4. When to Stop Testing? 251

11.5. Alternatives to Testing 252

11.6. Verification/Validation/Certification 252
11.6.1. Verification Techniques 253
11.6.2. Validation Techniques 254

11.7. Certification Process 255

11.8. Test Planning 257

11.9. Statistical Testing 257

11.10. Defect Testing 258

11.11. Stages in Testing Process 258
11.11.1. Unit Testing 258
11.11.2. Module Testing 258
11.11.3. Sub-System Testing 258
11.11.4. System Testing 259
11.11.5. Acceptance Testing 259

11.11.6. Beta Testing 259

11.12. Comparative Review of Testing Strategies 259
11.12.1. Top Down Testing 260
11.12.2. Bottom Up Testing 260
11.12.3. Thread Testing 261
11.12.4. Stress Testing 262
11.12.5. Back-to-Back Testing 262

11.13. Comparative Review of Defect Testing Approaches 264
11.13.1. Functional or Black-box testing 264
11.13.2. Structural or White-box testing 265
11.13.3. Interface Testing 266

11.14. Conclusions 268

12. FIELD DATA ANALYSIS 269

12.1. Introduction 269

12.2. Data Collection Principles 271
12.2.1. Study Plans, Goals and Input Variables 271
12.2.2. Failures, Faults, and Related Data 272
12.2.3. Time 274
12.2.4. Usage 275
12.2.5. Data Granularity 275
12.2.6. Data Maintenance and Validation 276
12.2.7. Analysis Environments 277

12.3. Data Analysis Principles 278
12.3.1. Plots and Graphs 279
12.3.2. Data Modeling and Diagnostics 282

12.4. Important Topics in Analysis of Field Data 282
12.4.1. Calendar Time 283
12.4.2. Usage Time 284
12.4.3. An Example 284

12.5. Calendar-Time Reliability Analysis 286

12.6. Usage-Based Reliability Analysis 287

12.7. Special Events 287
12.7.1. Rare Event Models 288
12.7.1.1. Constant Failure-Rate Model 288
12.7.1.2. Reliability Growth 289

12.8. Availability 290
12.8.1. Measuring Availability 290
12.8.1.1. Instantaneous Availability 290
12.8.1.2. Average Availability 291
12.8.2. Failure and Recovery Rates 291
12.8.3. Models 292
12.8.4. Prediction 293
12.8.5. Summary 293

13. STANDARDS AND HANDBOOKS 297

13.1. Reliability Standards & Handbooks 297
13.1.1. MIL-HDBK-H 108 Sampling Procedures and Tables for Life and Reliability Testing (Based on Exponential Distribution) 297
13.1.2. MIL-HDBK-189 Reliability Growth Management 297
13.1.3. MIL-HDBK-217F Reliability Prediction of Electronic Equipment 297
13.1.4. MIL-HDBK-251 Reliability/Design Thermal Applications 298
13.1.5. MIL-HDBK-263A Electrostatic Discharge Control Handbook for Protection of Electrical and Electronic Parts, Assemblies and Equipment (Excluding Electrically Initiated Explosive Devices) 298
13.1.6. MIL-HDBK-338 Electronic Reliability Design Handbook 299
13.1.7. MIL-HDBK-344 Environmental Stress Screening of Electronic Equipment 299
13.1.8. MIL-STD-690C Failure Rate Sampling Plans and Procedures 299
13.1.9. MIL-STD-721C Definition of Terms for Reliability and Maintainability 300
13.1.10. MIL-STD-756B Reliability Modeling and Prediction 300
13.1.11. MIL-HDBK-781 Reliability Test Methods, Plans and Environments for Engineering Development, Qualification and Production 300
13.1.12. MIL-STD-781D Reliability Design Qualification and Production Acceptance Tests: Exponential/ Distribution 300
13.1.13. MIL-STD-785B Reliability Program for Systems and Equipment, Development and Production 301
13.1.14. MIL-STD-790E Reliability Assurance Program for Electronic Parts Specifications 301
13.1.15. MIL-STD-1543B Reliability Program Requirements for Space and Missile Systems 301
13.1.16. MIL-STD-1629A Procedures for Performing a Failure Mode, Effects, and Criticality Analysis 302
13.1.17. MIL-STD-1686B Electrostatic Discharge Control Program for Protection of Electrical and Electronic Parts, Assemblies and Equipment (Excluding Electrically Initiated Explosive Devices) 302
13.1.18. MIL-STD-2074 Failure Classification for Reliability Testing 303
13.1.19. MIL-STD-2155 Failure Reporting, Analysis and Corrective Action System (FRACAS) 303
13.1.20. MIL-STD-2164 Environment Stress Screening Process for Electronic Equipment 303

13.2. Maintainability Standards & Handbooks 303
13.2.1. MIL-STD-470B Maintainability Program Requirements for Systems and Equipment 303
13.2.2. MIL-STD-471A Maintainability Verification/ Demonstration/ Evaluation 304
13.2.3. MIL-HDBK-472 Maintainability Prediction 304
13.2.4. DOD-HDBK-791 Maintainability Design Techniques 304
13.2.5. MIL-STD-1591 On Aircraft, Fault Diagnosis, Subsystems, Analysis/Synthesis of 305
13.2.6. MIL-STD-1843 Reliability-Centered Maintenance for Aircraft, Engines and Equipment 305
13.2.7. MIL-STD-2084 Maintainability of Avionic & Electronic Systems and Equipment 305
13.2.8. MIL-STD-2165A Testability Programs for Electronic Systems & Equipment 305
13.2.9. DOD-STD-1701 Hardware Diagnostic Test System Requirements 306
13.2.10. MIL-STD-2173 Reliability-Centered Maintenance Requirements for Naval Aircraft, Weapons Systems and Support Equipment 306
13.2.11. MIL-STD-001591A Subsystem Analysis/Synthesis of Command, Control & Communication (C3) System Component Fault Diagnosis 306

13.3. Safety Standards & Handbooks 306
13.3.1. MIL-HDBK-274 Electrical Grounding for Aircraft Safety 307
13.3.2. MIL-HDBK-764 System Safety Engineering Design Guide For Army Materiel 307
13.3.3. MIL-HDBK-828 Laser Range Safety 307
13.3.4. MIL-STD-882C System Safety Program Requirements 307
13.3.5. MIL-STD-1247C Markings, Functions and Hazard Designations of Hose, Pipe, and Tube Lines for Aircraft Missiles, and Space Systems 308
13.3.6. MIL-STD-1425A Safety Design Requirements for Military Lasers and Associated Support Equipment 308
13.3.7. MIL-STD-1576 Electroexplosive Subsystem Safety Requirements and Test Methods for Space Systems 308

13.4. Safety Standards & Handbooks **309**
13.4.1. MIL-HDBK-274 Electrical Grounding for Aircraft Safety 309
13.4.2. MIL-HDBK-764 System Safety Engineering Design Guide For Army Materiel 309
13.4.3. MIL-HDBK-828 Laser Range Safety 309
13.4.4. MIL-STD-882C System Safety Program Requirements 309
13.4.5. MIL-STD-1247C Markings, Functions and Hazard Designations of Hose, Pipe, and Tube Lines for Aircraft Missiles, and Space Systems 310
13.4.6. MIL-STD-1425A Safety Design Requirements for Military Lasers and Associated Support Equipment 310
13.4.7. MIL-STD-1576 Electroexplosive Subsystem Safety Requirements and Test Methods for Space Systems 311

13.5. Other Relevant Military Documents **311**
13.5.1. MIL-STD-105E Sampling Procedures and Tables for Inspection by Attributes 311
13.5.2. MIL-STD-337 Design To Cost 311
13.5.3. MIL-STD-454N Standard General Requirements for Electronic Equipment 312
13.5.4. MIL-HDBK-728 NonDestructive Testing (NDT) 312
13.5.5. MIL-HDBK-729 Corrosion and Corrosion Prevention Metals 312
13.5.6. MIL-HDBK-772 Military Packaging Engineering 313
13.5.7. MIL-HDBK-798 System Engineer's Design for Discard Handbook 313
13.5.8. MIL-STD-810E Environmental Test Methods and Engineering Guidelines 313
13.5.9. MIL-STD-883D Test Methods and Procedures for Microelectronics 314
13.5.10. MIL-STD-965B Parts Control Program 314
13.5.11. MIL-STD-975M NASA Standard Electrical, Electronic, and Electro Mechanical (EEE) Parts List 314
13.5.12. MIL-STD-1369 Integrated Logistic Support Program Requirements 315
13.5.13. MIL-STD-1388-1A Logistics Support Analysis 315
13.5.14. MIL-STD-1388-2B DOD Requirements for a Logistic Support Analysis Record 315
13.5.15. MIL-STD-1556B Government/Industry Data Exchange Program (GIDEP) 316
13.5.16. MIL-STD-1568B Materials and Processes for Corrosion Prevention and Control in Aerospace Weapons Systems 316
13.5.17. RAC NPRD Nonelectronic Parts Reliability Data, 1991 316

13.6. Non-Military Documents **317**

APPENDIX A - FIELD DATA **331**

1. Reliability Mathematics

This chapter presents a brief review of statistical principles, terminology and probability distributions used in the area of reliability. The objective of this chapter is to introduce concepts from probability and statistics that will be utilized in later chapters of this reference.

1.1. A Brief Introduction to Probability Theory and Statistics

1.1.1. Basic Definitions

Before considering the methodology for estimating system reliability, some basic concepts from probability theory must be reviewed. The terms that follow are important in creating and analyzing reliability block diagrams.

1. Experiment (E): An experiment is any well-defined action that may result in a number of outcomes. For example, the rolling of dice can be considered an experiment.
2. Outcome (O): An outcome is defined as any possible result of an experiment.
3. Sample space (S): The sample space is defined as the set of all possible outcomes of an experiment.
4. Event: An event is a collection of outcomes.
5. Union of two events A and B ($A \cup B$): The union of two events A and B is the set of outcomes that belong to A or B or both.
6. Intersection of two events A and B ($A \cap B$): The intersection of two events A and B is the set of outcomes that belong to both A and B.
7. Complement of event A ( ): A complement of an event A contains all outcomes of the sample space, S, that do not belong to A.
8. Null event ( ): A null event is an empty set and it has no outcomes.
9. Probability: Probability is a numerical measure of the likelihood of an event relative to a set of alternative events. For example, there is a 50% probability of observing heads relative to observing tails when flipping a coin (assuming a fair or unbiased coin).

1.1.2. Probability Properties, Theorems and Axioms

The probability of an event A is expressed as $P(A)$, and has the following properties:

1. $0 \leq P(A) \leq 1$.
2. $P(A) = 1 - P(\overline{A})$
3. $P(\emptyset) = 0$.
4. $P(S) = 1$.

In other words, when an event is certain to occur, it has a probability equal to 1; when it is impossible to occur, it has a probability equal to 0.

The probability of the union of two events A and B is:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Similarly, the probability of the union of three events, A, B and C is given by:

$$\begin{aligned} P(A \cup B \cup C) = {} & P(A) + P(B) + P(C) \\ & - P(A \cap B) - P(A \cap C) \\ & - P(B \cap C) + P(A \cap B \cap C) \end{aligned}$$

1.1.2.1. Mutually Exclusive Events

Two events A and B are said to be mutually exclusive if it is impossible for them to occur simultaneously. In such cases, the expression for the union of these two events reduces to the following, since the probability of the intersection of these events is defined as zero.

1.1.2.2. Conditional Probability

The conditional probability of two events A and B is defined as the probability of one of the events occurring knowing that the other event has already occurred. The expression below denotes the probability of A occurring given that B has already occurred.

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

1.1.2.3. Independent Events

If knowing B gives no information about A, then the events are said to be independent and the conditional probability expression reduces to:

$$P(A \mid B) = P(A)$$

From the definition of conditional probability, we can write:

$$P(A \cap B) = P(A \mid B)P(B)$$

Since events A and B are independent, the expression reduces to:

$$P(A \cap B) = P(A)P(B)$$

If a group of n events A_i are independent, then:

$$P\left[\bigcap_{i=1}^{n} A_i\right] = \prod_{i=1}^{n} P(A_i)$$

1.1.3. Random Variable

In general, most problems in reliability engineering deal with quantitative measures, such as the time-to-failure of a product, or qualitative measures, such as whether a product is defective or non-defective. We can then use a *random variable* X to denote these possible measures. In the case of times-to-failure, our random variable X is the time-to-failure of the product and can take on an infinite number of possible values in a range from 0 to infinity (since we do not know the exact time *a priori*). Our product can be found failed at any time after time 0 (*e.g.* at 12 hours or at 100 hours and so forth), thus X can take on any value in this range. In this case, our random variable X is said to be a *continuous random variable*. In this reference, we will deal almost exclusively with continuous random variables. In judging a product to be defective or non-defective, only two outcomes are possible. That is, X is a random variable that can take on one of only two values (let's say defective = 0 and non-defective = 1). In this case, the variable is said to be a *discrete random variable*.

1.2. The Probability and Cumulative Density (Distribution) Functions

The probability density function (*pdf*) and cumulative distribution function (*cdf*) are two of the most important statistical functions in reliability and are very closely related. When these functions are known, almost any other reliability measure of interest can be derived or obtained.

1.2.1. Designations

From probability and statistics, given a continuous random variable X we denote:

- The probability density (distribution) function, *pdf*, as *f(x)*.
- The cumulative distribution function, *cdf*, as *F(x)*.

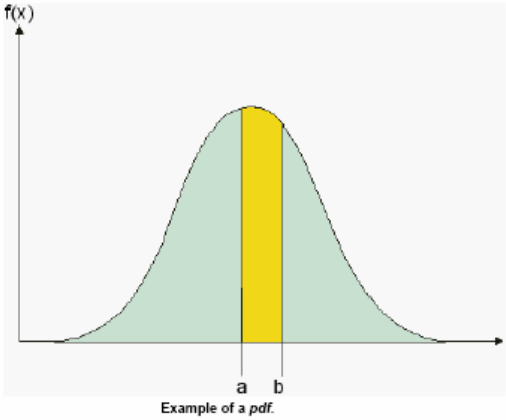
The *pdf* and *cdf* give a complete description of the probability distribution of a random variable.

1.2.2. Definition

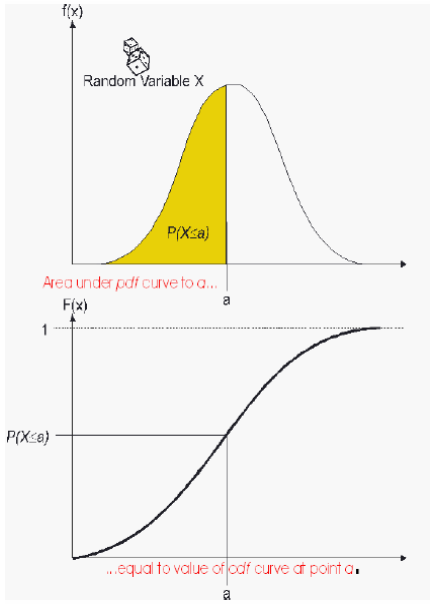
If *X* is a continuous random variable, then the *probability density function*, *pdf*, of *X* is a function, *f(x)*, such that for two numbers, *a* and *b* with *a* <=*b*:

$$P(a \le X \le b) = \int_a^b f(x)dx$$

That is, the probability that *X* takes on a value in the interval [*a*, *b*] is the area under the density function from *a* to *b* as shown in Figure below. The *pdf* represents the relative frequency of failure times as a function of time.



Example of a *pdf*.



The cumulative distribution function, cdf, is a function, F(x), of a random variable X, and is defined for a number x by:

$$F(x) = P(X \le x) = \int_0^x f(s)ds$$

That is, for a number *x*, *F(x)* is the probability that the observed value of *X* will be at most *x*. The *cdf* represents the cumulative values of the *pdf*. That is, the value of a point on the curve of the *cdf* represents the area under the curve to the left of that point on the *pdf*. In reliability, the *cdf* is used to measure the probability that the item

in question will fail before the associated time value, *t*, and is also called *unreliability*. Note that depending on the distribution function, denoted by *f(x)*, the limits will vary based on the region over which the distribution is defined.

1.2.3. Mathematical Relationship Between the pdf and cdf

The mathematical relationship between the *pdf* and *cdf* is given by:

$$F(x) = \int_0^x f(s)ds \quad \text{and} \quad f(x) = \frac{d(F(x))}{dx}$$

The total area under the *pdf* is always equal to 1, or mathematically:

$$\int_0^\infty f(x)dx = 1$$

1.2.4. Mean Life (MTTF)

The mean life function, which provides a measure of the average time of operation to failure, is given by:

$$\overline{T} = m = \int_0^\infty t \cdot f(t)dt$$

This is the expected or average time-to-failure and is denoted as the *MTTF* (Mean Time To Failure). (Note: Many practitioners and authors mistakenly refer to this metric as the *MTBF*, Mean Time Between Failures. The two metrics are identical if the failure rate of the component or system is constant. However, if the failure rate is not constant then the mean time to failure and the mean time between failures are not the same! Furthermore, *MTBF* only becomes meaningful when dealing with repairable systems.

NOTE: *The MTTF, even though an index of reliability performance, does not give any information on the failure distribution of the product in question when dealing with most lifetime distributions. Because vastly different distributions can have identical means, it is unwise to use the MTTF as the sole measure of the reliability of a product.*

1.2.5. Median Life

Median life, is the value of the random variable that has exactly one-half of the area under the pdf to its left and one-half to its right. It represents the centroid of the

distribution. The median is obtained by solving the following equation: (For individual data, the median is the midpoint value.)

$$\int_0^{\tilde{T}} f(t)dt = 0.5$$

1.2.6. Modal Life

The modal life (or mode), is the value of *T* that satisfies:

$$\frac{d[f(t)]}{dt} = 0$$

For a continuous distribution, the mode is that value of *t* that corresponds to the maximum probability density (the value at which the *pdf* has its maximum value, or the peak of the curve).

1.3. Statistical Distributions used in Reliability Analysis

The following distributions are considered to be the most valuable tools for reliability analysis:

- **Exponential**
- **Binomial (Discrete)** This distribution is very useful for Quality Assurance and Reliability modeling. It is applied in situations where the two events are complementary, such as good or bad, success or failure, working or not working, etc. Restricted to finite sample size. This distribution is very useful in reliability studies for computing the probability of success when the system employs partial redundancy. It has many applications in Reliability & Maintainability analysis for estimating the number of spares and logistic support based on the probability of failures. It cannot be directly applied to calculate event probabilities in the time domain, as the total number of occurring and non-occurring events is usually unknown. If the precise detail of failures and successes is not easily available for the given time, the application of Binomial Distribution becomes difficult.
- **Poisson (Discrete)** It is a special case of the Binomial Distribution when p is very small and n very large, the limiting form of this distribution is called the Poisson distribution. The Poisson is an extension of the binomial distribution in which the number of samples is infinite. For this distribution all you need to know is the value of r, the expected number of failures and

you can find all the probabilities of various occurrences without knowing the number of trial or sample size. The probability of no failures in time t is given by the first term, exp (-λt) of the above series, which is the Reliability by definition. The second term λt. e-λt is the probability of exactly one failure, and so on. This means that from knowing the failure rate, λ the reliability or unreliability of an item can be determined for any length of time. This is one of the most remarkable properties of Poisson distribution, which overcomes the difficulties of calculating event probabilities in the time domain as seen in the case of a Binomial. The events following Poisson distribution occur at a constant average rate and the number of events occurring in any interval of time is independent of the number of events occurring in any time interval. Often used to describe situations in which the probability of an event is small and the number of opportunities for the event is large. Good for the inspection sampling plans. It can also be used in certain cases of redundant configurations of complex systems for calculating the partial unreliabilities. It is most commonly used for calculating the number of spare items for operation and logistic support of a system Poisson distribution is a very useful tool for modelling the demand and spare capacity of telephone lines at the Distribution Points (DP). The following figure depicts a pattern of spares lines and demand for a typical set of DPs associated with a Telephone Exchange.

- **Normal**
- **Log-normal**
- **Weibull**
- **Chi-square** This distribution is quite often used in Hypothesis testing for data analysis. It is also used to determine the confidence intervals for the MTTF or MTBF The Chi-Square distribution is a good tool for counting the number of failures in a given interval.
- **Beta** Beta functions are used in reliability for the development of ranking distributions when used in the context of life testing.
- **Gamma** This distribution is applied in reliability analysis where a given number of partial failures must occur before an item fails completely, for

instance, in redundancy analysis It is also used to describe an increasing or decreasing hazard function The gamma distribution can also be applied to model the time to the nth failure of a system if the underlying failure distribution is Exponential.

- **Erlang** This distribution is commonly used in the Renewal Theory for determining the ample supply of spares for repairable systems and to schedule the workload of repairmen.

- **Extreme Value** This distribution is used in Reliability analysis of mechanical devices, for example, failures caused by corrosion. It is used for fitting the limiting distribution for the maximum number of samples collected from a process. This distribution applies when extremes rather than means are collected from samples from an unknown or complex underlying distribution.

- **The Classical Bathtub Distribution** The most useful function is the Hazard Function for the classical bathtub, which can be expressed as: **$h(t) = \beta\eta\ (\eta t)\beta\text{-}1\ \exp(\eta t)\beta$ for $t \geq 0$, β (Shape Parameter)>0, and η(Scale Parameter) >0.** The classical Bathtub distribution is used to represent a trimodal failure distribution of items, which follow decreasing, increasing or constant hazard rates

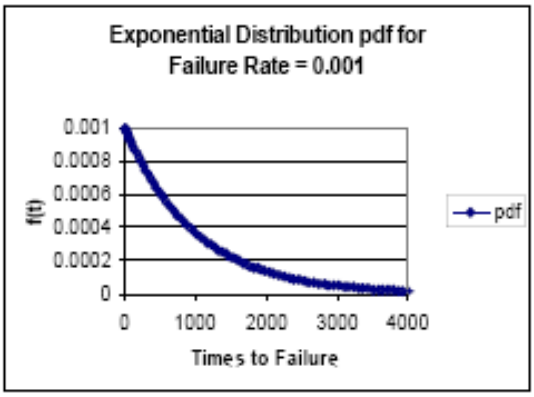
1.3.1. Exponential Distribution

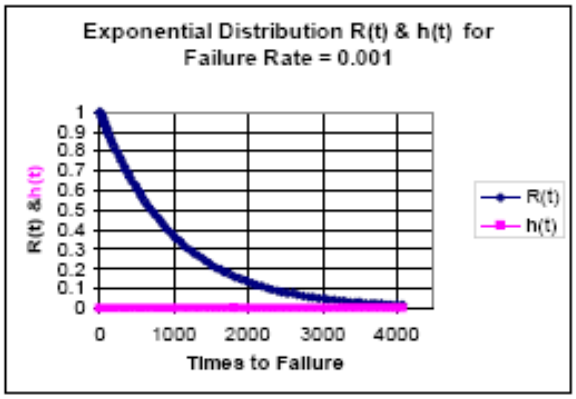
This is the most important distribution in Reliability Theory. When an item is subject to failures, which occur randomly, the probability that the item will not fail within a given interval of time is a simple exponential function of that time interval. This statement is subject to the following conditions:

- that the item has survived to the beginning of the time interval

- the age of the item is such that it does not reach the end of its life within that interval. The exponential failure density function is expressed as $f(t) = \lambda \exp(-\lambda t)$, $t >= 0$ where λ is the failure rate. The Reliability Function for the exponential distribution is given by $R(t) = \exp(-\lambda t)$, $t >= 0$ And the Hazard Function is deduced simply as $h(t) = \lambda$. The following diagram depicts the

shapes of f(t), R(t) and h(t) for an item with a failure rate of 0.001 units of time.





Some of the fundamental properties of exponential distribution:

1. The single parameter, λ completely determines the reliability of an item
2. This distribution is independent of the age on an item
3. A population of items following an exponential failure distribution suffers its greatest failures, nearly 63%, in the period less than its MTBF, provided the failed items are not replaced
4. The total area under the exponential density curve is unity
5. The exponential distribution is additive. This implies that the sum of a number of exponentially distributed variables is also exponentially distributed
6. The mean Life for a repairable item following exponential distribution is given by the expression **Mean Life = 1/λ = MTBF**

Some typical applications of exponential distribution in Reliability Practice

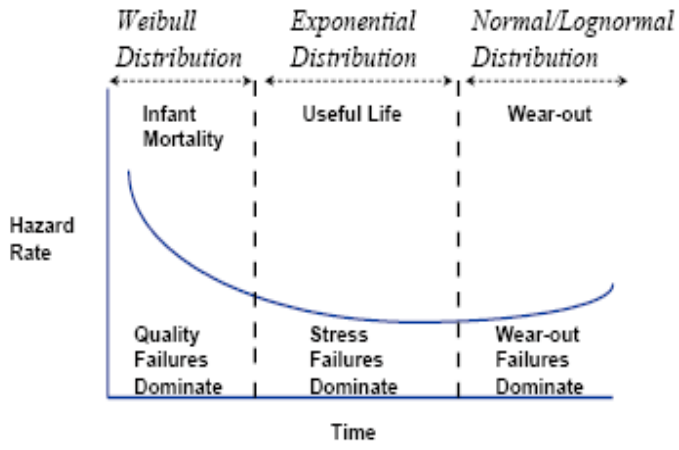
The exponential distribution is a very useful model for reliability analysis of items, which exhibit a constant failure rate during their operation life.

- The total failure rate of a number of statistically independent items connected in series is simply the sum of the constant failure rates of individual items. This principle is applied to estimate the MTBF of a product (without redundancy) by simply adding the individual component failure rates and inverting the total failure rate. Hence, the technique is called Reliability Prediction by Parts Count.

- This distribution is almost exclusively used for predicting the reliability of electronic products for which the infant mortality failures have been screened

out and the wear out failures are prevented by timely replacement of failed items/components

- The exponential distribution model is ideal for testing a product, which behaves exponentially, due to the fact that a single parameter λ can determine the reliability uniquely and completely. All that is required is to determine the value of λ by conducting a test or a field trial. Other distributions require more than one parameter to be determined.
- The expression **MTBF = 1 / λ** is only true for the exponential distribution, and generally holds true for the useful life of an item where random failures dominate, as shown in the Bathtub curve in Figure below.



1.3.2. The Normal Distribution

The Normal or Gaussian distribution is the best-known two-parameter distribution. It was first discovered by De Moivre in 1733, but somehow it has been attributed to Gauss and hence the Gaussian Distribution. This distribution is very often a good fit in many situations, particularly, when a parameter, which is a random variable is the sum of many other random variables, then the parameter will exhibit a normal distribution in most of the cases. For example, the variations in the values of electronic parts due to manufacturing are considered to be normally distributed about a mean value of the parameter (size or weight, etc.) being measured. The fundamental basis of this is the Central Limit Theorem, which states that the sum of a large number of independently distributed random variables each with a finite mean and standard deviation is normally distributed. There are several applications of this distribution in Quality, Reliability and Maintainability analysis. The normal failure density function used for describing wear-out failures is given by

$$f(T) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-(T-\mu)^2 / 2\sigma^2]$$

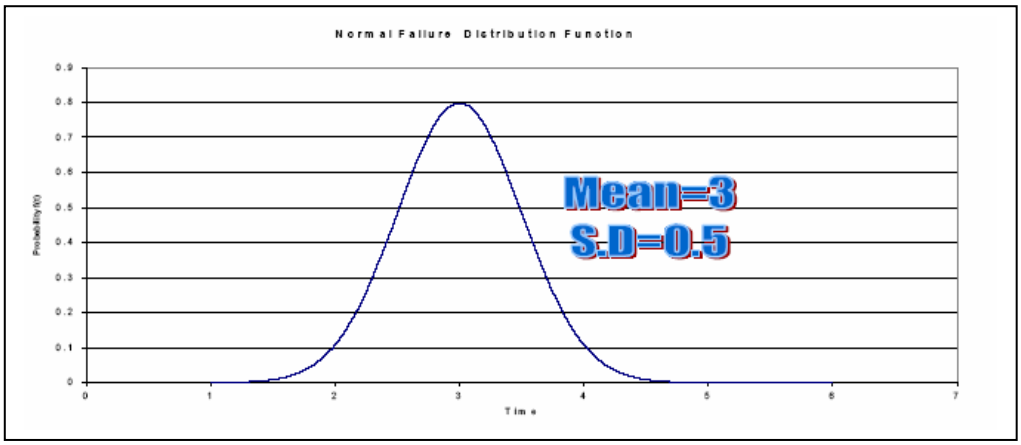
Where, T = is the age of the item, and μ = mean wear-out life and σ , the S.D of the item lifetimes from the mean μ. It is important to remember that this distribution depends upon the age of the item. When the item age equals the mean wear-out life i.e., when T = μ then the above expression reduces to

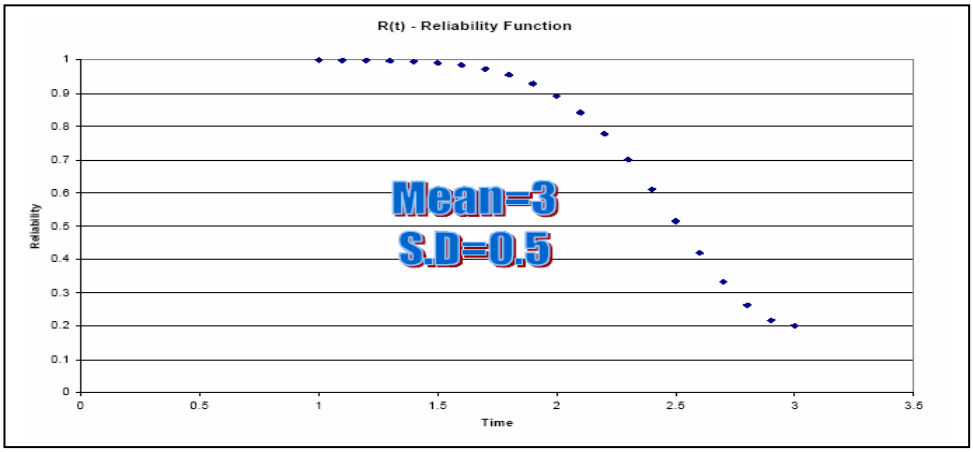
$$\mathbf{f(\mu) = 1/\sigma\sqrt{2\pi} \quad = 0.399/\sigma}$$

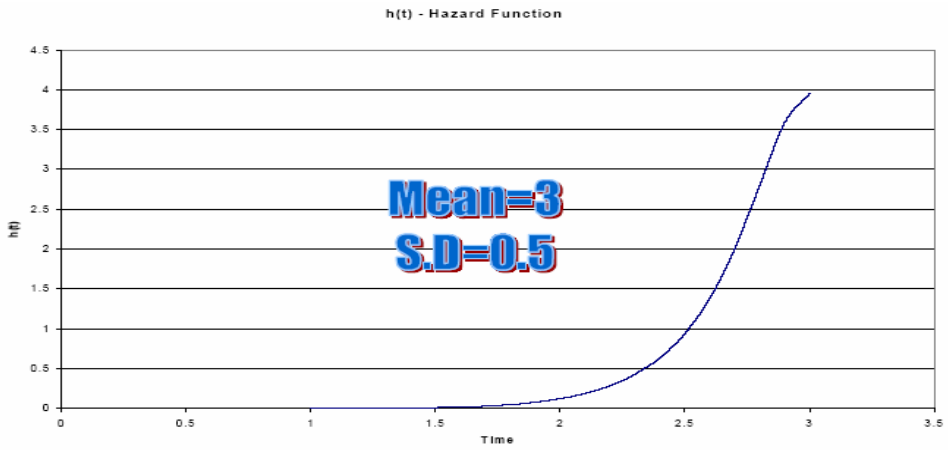
This gives ***the probability of failure at the mean wear-out life*** of the item. The reliability function R(T) is given by

$$R(T) = \int_{T}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp[-(T-\mu)^2 / 2\sigma^2] \; dT \text{ and}$$

The Hazard Function, **h (T) = f(T)/R(T),** which is a monotonically increasing function of T. The normal failure distribution, reliability and hazard functions for a for a Mean of 3 and Standard Deviation of 0.5 are shown in the following figures.







Some fundamental properties of Normal Distribution

- It is a continuous distribution
- The mean, median and mode of a perfectly normal distribution are all equal
- A perfectly normal distribution has zero coefficient of Skewness and the coefficient of Kurtosis is 3.
- A population of items conforming to a perfectly normal distribution is symmetrically dispersed about the mean
- Since the two tails of a normal distribution are symmetrical, a given spread includes equal values in each tail
- A normal distribution can be evaluated from the standardized normal distribution by using the transformation, $\mathbf{z = (x - \mu)/ \sigma}$
- The normal distribution represents a limiting case of the Binomial and the Poisson distributions; hence it can be used to provide a good approximation for these distributions when the number of items in a sample is large
- The normal distribution depends on the age of an item

Particular Applications of the Normal Distribution

- One of the main applications of this distribution is in reliability analysis of items, which exhibit wearout failures, for instance, mechanical and electromechanical devices
- The other principal application deals with the analysis of manufactured products and their ability to meet specified requirements

- It is quite frequently used in quality control procedures and analyzing the strength of materials and components
- The normal distribution represents the wearout phenomena quite well. It can be seen that about half the failures occur before the start of Mean Life and the other half occur later. In the case of a normal distribution the failures cluster around the mean life. This implies that the failure free operation can often be achieved up to the age of an item relatively close to the mean life of the item, according to how widely the curve is spread
- The reliability of an item is given by R (MTBF) = 0.5 for a Normal distribution, which is symmetrical about its mean.
- This distribution is vital importance for the maintenance engineer to carry our wearout studies for the establishment of a preventive maintenance strategy for long-life equipment. It is also applied for assuring that the wearout phenomena cannot affect a one-shot device during its critical mission.

1.3.3. The Log-Normal Distribution

The Log-Normal distribution is the distribution of a random variable whose natural logarithm is normally distributed; in other words, when working with the Log-Normal distribution just change the values of the random variable, for instance, time 't' to log(t) as normally distributed. The pdf of a Log-Normal distribution is described by the expression

$$f(x) = (1/\sigma x\sqrt{\pi}).\ \exp\left(-\{\ln(x-\mu)^2/2\sigma^2\}\right) \text{ for } x > 0 \text{ and}$$

$f(x) = 0$ for $x < 0$; where μ = Mean and σ = Standard Deviation of ln(x).

This distribution has the cdf given by the following integral

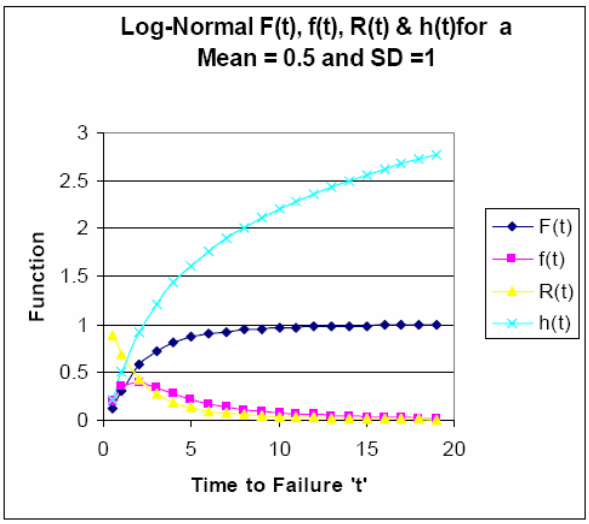
$$F(x) = \int_0^x (1/\sigma x\sqrt{\pi}).\ \exp\left(-\{\ln(x-\mu)^2/2\sigma^2\}\right) dx \text{ for } x > 0$$

and the Reliability Function is given by the integral

$$R(t) = \int_t^\infty (1/\sigma x\sqrt{\pi}).\ \exp\left(-\{\ln(x-\mu)^2/2\sigma^2\}\right) dx$$

The Hazard Function is then deduced by: $h(t) = f(t)/R(t)$

The cdf, pdf, Reliability and Hazard Functions for a Log-Normal distribution of times to failure 't' with a Mean of 0.5 and Standard Deviation of 1 are shown in Figure below:



Fundamental Properties of the Log-Normal Distribution

- This is a continuous distribution
- This distribution is more versatile than the Normal as it has a range of shapes. In the case of Normal distribution, the lower limit will have to be $-\infty$, but this doesn't make sense in practice. The difficulty is that the area under the Normal distribution curve becomes 'unity' only if the curve is extended to infinity in both directions. This is not possible for time dependent events where a new item enters the service at time zero. This difficulty is overcome by the fundamental property of the Log-Normal distribution as it has the advantage of having the value $f(x) = 0$ for $x = 0$.
- For scale parameter, the median, $m > 0$ and the mean, $\mu > 0$ the following relationship is quite useful $m = \exp(\mu)$ and $\mu = \log(m)$
- The graph of the function $\log[f(x)]$ against $\log(x)$ depicting a straight line is a test for a perfect fit of Log-Normal distribution.
- When $\mu >> \sigma$, the Log-Normal distribution approximates to the Normal.

Particular Applications of the Log-Normal Distribution

- This distribution is applied in the situations where the hazard rate function increases to a maximum value and then decreases with time.

- It is most frequently used to describe the behavior of mechanical and electromechanical devices and to determine the start of wearout phenomena and to calculate the wearout failure rate.
- This distribution like the Normal distribution depends on the age of an item. This is the basis on its main application in maintainability analysis.
- A population of items behaving Log-normally, when put on a test trial where the failed items are not replaced, suffers its greatest failures around the mean-life.

1.3.4. The Weibull Distribution

The Weibull distribution is the complex of the distributions most frequently used in reliability analysis. It is a more general three-parameter distribution and other distributions, such as, Exponential, Normal, Log-normal, Gamma and Rayleigh distributions are special cases of this distribution. The Weibull failure density function is associated with the times to failure of items and it is uniquely defined by three parameters. By adjusting the Weibull distribution parameters, it can be made to model a wider range of applications. The general form of density function for a three-parameter Weibull distribution is given by:

$$f(t) = \frac{\beta\,(t-\gamma)^{\beta-1}}{(\eta-\gamma)^{\beta}} \exp\left[-\left\{\frac{(t-\gamma)^{\beta}}{(\eta-\gamma)^{\beta}}\right\}\right] \text{ for } t \geq \gamma \geq 0$$

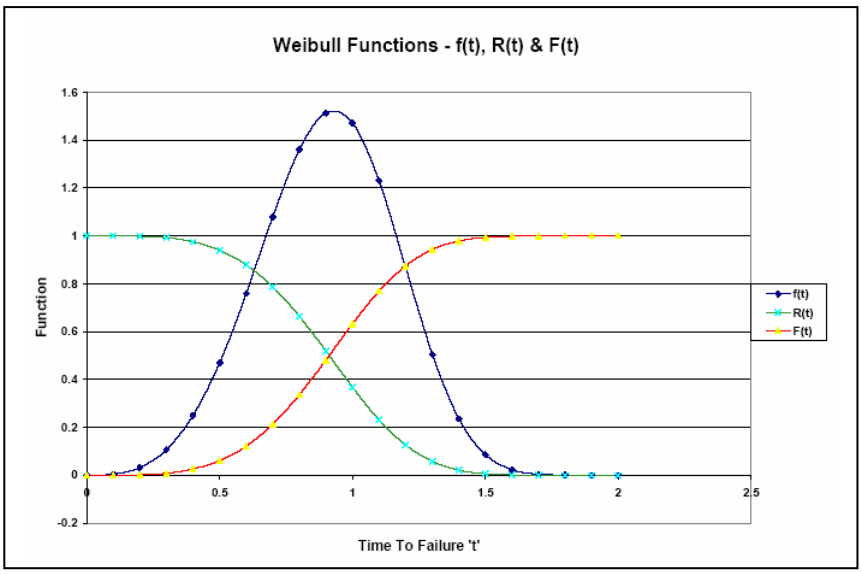
Where, β is called the Shape Parameter, η is the scale parameter, which is also called the Characteristic Life at which about 63% of the population of items would have failed. The third parameter γ is called a Location Parameter or minimum life. The Reliability Function for this distribution for $t >= \gamma$ is given by the expression,

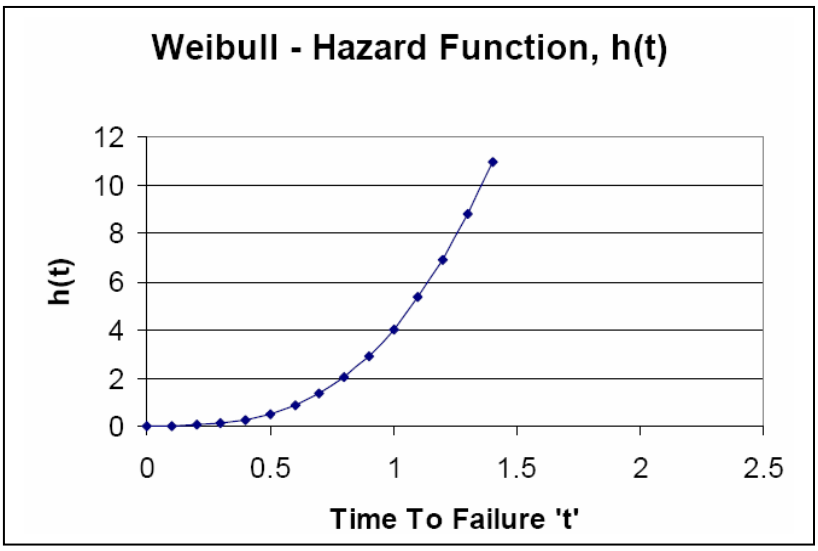
$$R(t) = \exp\left[-\left\{\frac{(t-\gamma)^{\beta}}{(\eta-\gamma)^{\beta}}\right\}\right]$$

$$h(t) = f(t)/R(t) = \left[\beta\,(t-\gamma)^{\beta-1}\right] / \left[(\eta-\gamma)^{\beta}\right], \text{ for } t \geq \gamma \geq 0$$

This distribution is very flexible and using different values of the three parameters it can depict various shapes of the above functions. As an example, the shapes of the failure Density Function, Reliability Function, Cumulative Density

Function and Hazard Function have been for the following values of the three parameters, β **= 4,** η **= 1 and** γ **= 0.**





The most commonly used density function for a Weibull distribution is given by the following simplified expression where the term (η-γ) is considered as a scale parameter, which is always positive:

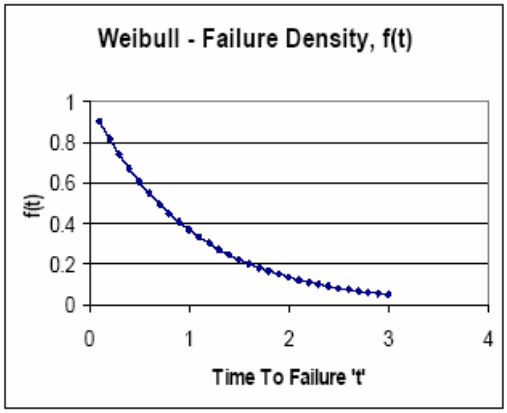
$$f(t) = \frac{\beta}{\eta}\left\{\frac{t-\gamma}{\eta}\right\}^{\beta-1} \exp[-(\frac{t-\gamma}{\eta})^{\beta}] \text{ for } t \geq \gamma \geq 0$$

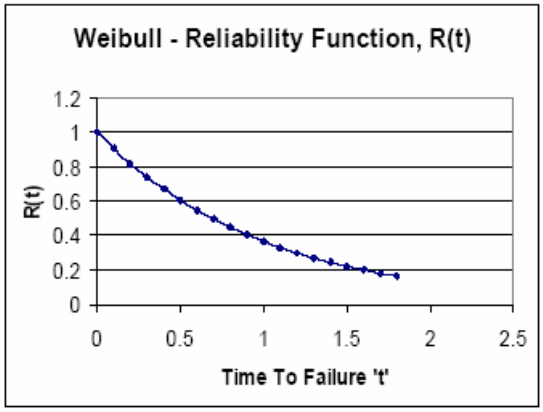
$$R(t) = \exp[-(\frac{t-\gamma}{\eta})^{\beta}] \text{ for } t > 0$$

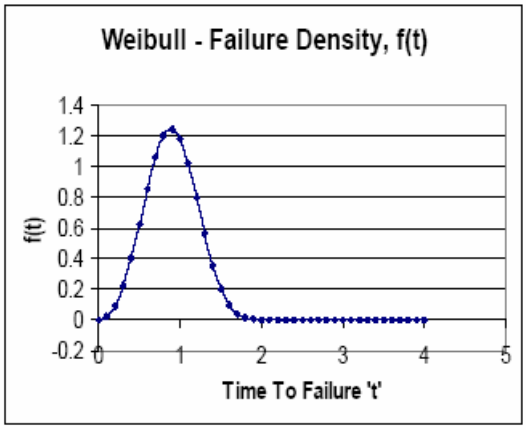
In most of the practical applications the failures are assumed to start at time zero, which implies that the location parameter, γ =0. And substituting 0 for γ can further simplify the above expressions.

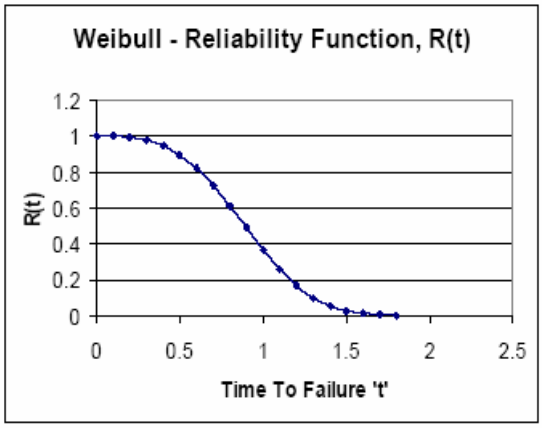
The Exponential distribution is a particular case of the Weibull with $\beta = 1$, $\gamma = 0$. In this case,

$f(t) = (1/\eta) \exp -(t/\eta)$ and $R(t) = \exp -(t/\eta)$, where $(1/\eta)$ corresponds to the constant failure rate λ. If the value of $\beta = 3.2$ and $\gamma = 0$, the Weibull distribution approximates closely to the Normal distribution where η corresponds to the mean life and β to the standard deviation. Figures depict the shapes of R(t) for the above two cases.









Some fundamental properties of Weibull Distribution

- It is a continuous distribution
- This distribution is associated with times to failure of items and it supplements the Exponential and the Normal distributions.
- While the Exponential is described by a single parameter and the Normal described by two parameters, three parameters are required to uniquely describe the Weibull.

- A three-parameter Weibull distribution can be reduced to a two-parameter distribution by assuming that the location parameter γ is always zeroed.
- It is a more general three-parameter distribution and other distributions, such as Exponential, Normal, Log-normal, Gamma and Rayleigh distributions are special cases of this distribution.
- Depending upon the value of the shape parameter, the Weibull distribution shows the following properties:

β-Value	Distribution Type	Hazard Rate
β<1	Gamma	Decreasing
β=1	Exponential	Constant
1<β<2	Log-Normal	Increasing
β=2	Rayleigh	Increasing at a linear rate
β=3.2	Normal	Increasing/decreasing

- The Weibull probability graph paper is particularly useful as an exploratory technique in understanding life test or field data from a product.

Particular Applications of the Weibull Distribution

- This distribution holds an important place among the distributions of lifetime due to the fact that a small difference in the distributions of lifetime of components can describe the lifetime of a product. For, example, if each of the components has a normal distribution of lifetime but the parameters of these distributions vary somewhat from component to component, then for a sufficiently large number of components, the Weibull is the best distribution to apply.
- The infant-mortality and wearout failure mechanisms are best described by this distribution. The values of the three parameters of the Weibull distribution can be determined from test data or field data using Maximum Likelihood Estimation (MLE) technique. The estimated values of these parameters can indicate a number of things about the product's life cycle-**If β < 1 then h(t) will decrease with time ,t (Represents Early life) If β = 1 then h(t) will be constant with time ,t (Represents Useful life) If β > 1 then h(t) will increase with time ,t (Represents Wear-out)**
- The estimated value of the location parameter (γ) indicates the following situations: A value of less than zero indicates failure in storage. These failures end up as Dead On Arrival (DOA) when a batch of items is delivered. A

positive value of the location parameter suggests that there is some period of time, which is failure free. This could be considered as a failure free warranty period.

- It is clear from the above discussion that the Weibull distribution can be applied to model a variety of situations by the right choice of the parameter values. The Weibull is particularly useful in reliability work due to its flexibility to model a wide range of lifetime distributions of different items.

2. Burn In

2.1. Introduction

Reliability engineers have long recognized an inherent characteristic in many types of equipment to exhibit a decreasing failure rate during their early operating life. Intuitively, a reliability high early failure rate that decreases with time until it eventually levels off can be explained by the inherent variability of any production process.

The 'substandard' portion of the production of identical parts can be expected to fail early, and they do so quickly. The failures of these substandard parts are labeled "early life failures." Experience shows that semiconductors, prone to fail early, will usually fail within the first 1,000 operating hours under use conditions. After that the failure stabilizes, perhaps for as long as 25 years, before beginning to increase again as the components go into wear out. These failures, termed "infant mortalities," can be as high as 10% in a new, unproven technology and as low as 0.01% in a proven technology.

Burn-in test assures that substandard components, which do not meet their failure rate, mean life or reliability goal, are identified by subjecting them to high temperature, and at times in conjunction with other high stresses such as voltage, wattage, vibration, etc. this temperature and the additional stresses are higher than use condition stresses, and usually near their rated capacity or higher, but preferably not in excess of 20% above their capacity.

2.1.1. Burn-In Definitions

In MIL-STD-883, Method 1015.3, "Burn-in Test," burn-in is defined as follows:

Burn-in is a test performed for the purpose of screening or eliminating marginal devices, those with inherent defects or defects resulting from manufacturing aberrations which cause time and stress dependent failures.

Kuo and Kuo define the burn-in in as follows:

Burn-in is a stress operation that combines the appropriate electrical conditions with appropriate thermal conditions to accelerate the aging of a component or device. In other words, burn-in is a process which operates electronic components or systems under electrical and thermal conditions to demonstrate the real life of the components or systems in a compressed time.

2.1.2. The Differences between Burn-In and Environmental Stress Screening (ESS)

There have been a lot of confusions both in the industry and in the literature about the terms "Burn-in" and "ESS." "Burn-in" and "ESS" have been used interchangeably by many people. In fact, they are two very relevant but different concepts.

Burn-in is a generally lengthy process of *powering* a product at a specified *constant* temperature.

ESS evolved from burin-in techniques but is a considerably advanced process. Generally, ESS is an accelerated process of stressing a product in continuous cycles between predetermined environmental extremes, primarily temperature cycling plus random vibration.

The misconception is due to the wrong assumption that historical "burn-in" procedures conducted on electronic equipment, currently in the inventory, are as cost effective as the ESS temperature cycling and random vibration screens. illustrates the differences between burin-in and ESS procedures. It may be seen that burn-in can be regarded as a special case of ESS where the temperature change rate for thermal cycling is zero and vibration is sinusoidal if ever used.

2.2. Burn-In Methods and Their Effectiveness

2.2.1. Static Burn-In

The simplest type of burn-in is static, or steady-state, burn-in. Static burn-in maintains a steady-state bias on each device under high temperature for a number of hours to accelerate the migration of impurities to the surface so that a potential failure will occur. A static system is cheaper and simpler, and is useful with contamination-related failure mechanisms. It is, however, less effective than dynamic

burn-in for large scale integration (LSI) and very large scale integration (VLSI) devices.

Table 2.1: Comparison of burn-in test and ESS procedure

Criteria	Burn-in	ESS
Temperature	Operating or accelerated	Cycled from high to low operating
Vibration	Usually constant, but sometimes cycled	Random, normally 20-2,000 Hz
Temperature rate of change		5°C per minute minimum
Length of time	Normally 168 hours or less	10 or 5 minutes perpendicular to each axis of orientation for vibration, and 10 to 20 cycles for temperature cycling.

2.2.2. Dynamic Burn-in

Dynamic burn-in uses power source voltage, clock signals, and address signals to ensure all internal nodes are reached during temperature stressing. It is more effective at detecting early failures in complex device. It is also more expensive and requires more dedicated burn-in boards. It should be noted that a dynamic burn-in system can also be used for static burn-in, but not the other way around.

2.2.3. Test During Burn-In

A subset of dynamic burn-in is Test Burn-In (TDBI). TDBI adds functional testing and, possibly, monitoring of component outputs to show how they are responding to specific input stimuli. TDBI is the most comprehensive burn-in technique, especially when coupled with scan-based technology. It has been used primarily for dynamic random access memories (DRAMs) but is applicable for all large memories due to their long electrical test times. Normally, the electrical testing is performed after burn-in to detect failures. TDBI is not appropriate for EPROM'S (erasable programmable ROM's), microprocessors and other VLSI circuits.

A typical TDBI is performed in the following manner:

1. The devices are operated at an elevated temperature (125°C) and voltage (7 to 7.5 V) for an extended period of time, while all devices-under-test (DUT's) are subjected to function testing using a complex test pattern
2. The DUT's are operated for a short duration at a lower temperature (70°C) and voltage (5.5V) during which parametric testing is performed.
3. Repeat Steps 1 and 2 for 4 to 8 hours or longer.

2.2.4. High-Voltage Stress Tests

High-voltage stress tests are categorized as burn-in screens because of their device-aging-acceleration features due to the application of voltage, time and temperature. For these tests, the distribution of the voltage stress throughout the IC is accomplished by carefully designed dynamic and functional operation. IC memory suppliers have used high-voltage stress tests in lieu of dynamic burn-in as a means to uncover oxide defects in MOS IC's. Some suppliers use high-voltage stress tests in conjunction with either dynamic burn-in or TDBI.

A typical high-voltage stress test involves cycling through all addresses (for a memory) using selected memory data patterns for 2 seconds, both high and low (logically speaking), with 7.5 V forcing function being applied (for a 5-V rated part, for example).

Note that for VLSI devices which have a unique set of characteristics significantly different from small-scale integrated (SSI), medium-scale integrated (MSI) and large-scale integrated (LSI) devices, burn-in needs to be refined since both stress coverage and test coverage are required to develop an effective burn-in method for these devices.

Burn-in methods are generally classified into the following categories:

1. Elevated temperature plus power-the cheapest but the least effective method.
2. Elevated temperature plus power with all inputs reverse biased, or the so-called High Temperature Reverse Bias (HTRB) - a method with moderate cost and reasonable effectiveness for most devices.
3. Elevated temperature, power, dynamic excitation of inputs, and full loading of all outputs-an effective and expensive method.
4. Optimum biasing combined with temperature in the range of 200 to 300°C, or the so-called High-Temperature Operating Test (HTOT)- an expensive and difficult-to-carry-out method which is not applicable to plastic devices due to the high temperature involved.

No matter how the burn-in methods are classified, one thing is certain. Each failure mechanism has a specific activation energy that dominates the effectiveness of

each burn-in method. Burning-in components, by applying high voltage to their pins, accelerates the time-to-failure of oxide defects (weak oxide, pin holes, uneven layer growth, etc.) typically found in MOS devices. High temperature also accelerates these and other defects, such as ionic contamination and silicon defects. Table 2.2 is a summary of failure mechanisms accelerated by various popular burn-in methods based on the activation energies for these failure mechanisms. Table 2.3 is a summary of the effectiveness for these burn-in methods versus the major technology categories.

Table 2.2: Failure mechanisms accelerated by various burn-in methods

Failure mechanisms	Static burn-in forward bias	Static burn-in reverse bias	Dynamic burn-in pulsed	High voltage stress burn-in	TDBI	TDBI plus high voltage
Leakage current	•	•		•		
Surface instability/ ionic contamination		•		•		
Step coverage	•		•			
Oxide defects		•	•	•		•
Stress coverage					•	•
Epitaxial & crystal defects			•			
Metalization defects	•		•			
Junction anomalies	•		•			
Inversion		•				
Channeling		•				
Wire bond problems	•		•			
Tunneling				•		
Metal migration	•			•		
Refresh problems for DRAM (charge storage)				•†		
GaAs ohmic contact resistance (such as Au/Ag /Ni/AuGe/Ni contacts)		•				

† Effective at low temperature.

Table 2.3: Effectiveness of various Burn-In methods vs. technology categories

Technology categories	Static burn-in forward bias	Static burn-in reverse bias	Dynamic burn-in pulsed	High voltage stress burn-in	TDBI	TDBI plus high voltage
Linear ICs						
Contamination		•				
Step coverage /leakage	•					
CMOS logic						
Contamination		•				
Step coverage /leakage	•					
TTL logic (Std, S, H, LS, ALS, AS, F)	•					
Memories						
All			•		•	
MOS			•	•	•	•†
Microprocessors			•	•		
GaAs digital ICs:						
(1) SSI	•					
(2) MSI, LSI, & RAM			•			

2.3. Burn-In Documents

Several military documents define burn-in standards which have been used throughout the industry. Among these, MIL-STD-781, issued in 1967, has been used to demonstrate the reliability of production electronic equipment. This standard has needed substantial improvements in the area of burn-in application as pointed out by many researchers. Consequently, a revised version of MIL-STD-785 lists burn-in as one of the eight (8) major tasks comprising a reliability improvements. In addition, MIL-STD-883 and MIL-HDBD-217 have become a focal point in system concepts for ensuring a successful reliability demonstration test. Industry generally accepts MIL-STD-883 as the basis for most burn-in conditioning done by manufacturers of industrial electronic equipment. However, there are some criticisms concerning its ineffectiveness, expensiveness and possible damages to the equipment [8]. Also, there is little adequate theory to permit the calculation of the optimum burn-in time.

2.4. Burn-In Test Conditions Specified By MIL-STD-883C

In MIL-STD-883C, the following six basic test conditions are specified for the burn-in test:

1. **Test Condition A- Steady Sate, Reverse Bias**

   This test condition is illustrate in Figure 2.1 and is suitable for use on all types of circuits, both linear and digital. In this test, as many junctions as possible will be reverse biased to the specified voltage.

2. **Test Condition B – Steady State, Forward Bias**

   This test condition is also illustrated in Figure 2.1 and can be used on all digital type circuits and some linear types. In this test, as many junctions as possible will be forward biased to the specified voltage.

3. **Test Condition C – Steady State, Power and Reverse Bias**

   This test condition is also illustrated Figure 2.1 in and can be used on all digital type circuits and some linear types where the inputs can be reverse biased and the output can be biased for maximum power dissipation or vice versa.

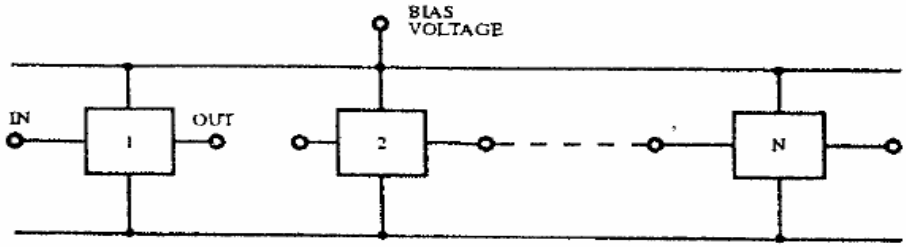


Figure 2.1: Steady state burn-in test for test condition A, B, and C

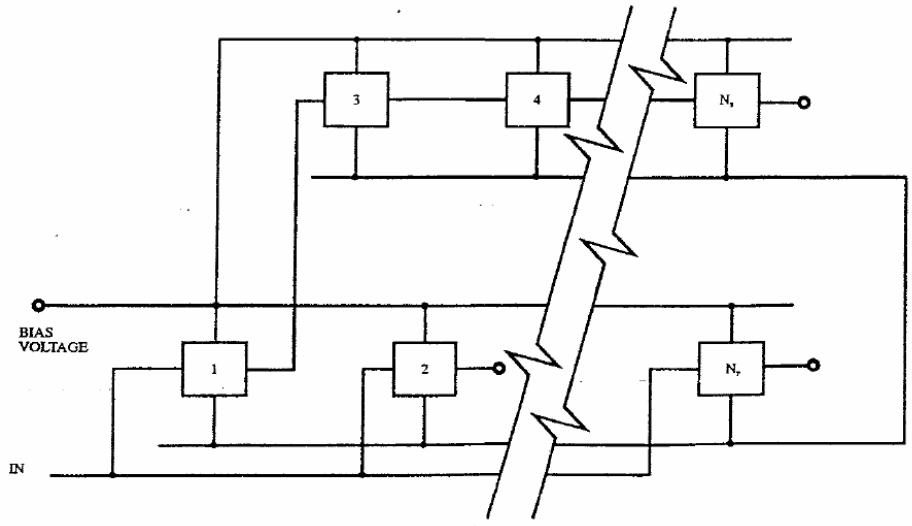


Figure 2.2: Typical parallel, series excitation for test condition D

4. Test Condition D – Parallel or Series Excitation

This test condition is typically illustrated in Figure 2.2 and is suitably used on all circuit types. Parallel or series excitation, or any combination thereof, is permissible. However, all circuits must be driven with an appropriate signal to simulate, as closely as possible, circuit application and all circuits should have the maximum load applied. The excitation frequency should not be less than 60 Hz.

5. Test Condition E – Ring Oscillator

This test condition is illustrated in Figure 2.3, with the output of the last circuit normally connected to the input of the first circuit. The series will be free running at a frequency established by the propagation delay of each circuit and associated wiring, and the frequency shall not be less than 60 Hz. In the case of circuits which cause phase inversion, an odd number of circuits shall be used. Each circuit in the ring shall be loaded to its rated maximum capacity.

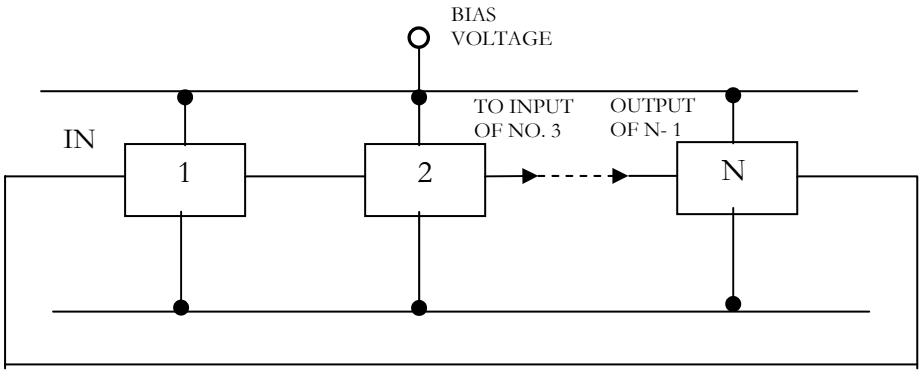


Figure 2.3: Ring oscillator for Test Condition E

6. Test Condition F – Temperature – Accelerated Test

Under this test condition, microcircuits are subjected to bias(es) at an ambient test temperature, typically from 151°C to 300°C, which considerably exceeds their maximum rated junction temperature. It is generally found that microcircuits will not operate properly at these elevate temperatures in their applicable procurement documents. Therefore, special attention should be given to the choice of bias circuits and conditions to assure that important circuit areas of the circuit. To properly select the accelerated test conditions, it is recommended that an adequate sample of devices be exposed to the

intended high temperature while measuring voltage(s) and current(s) at each device terminal to assure that the applied electrical stresses do not induce damaging overstresses. Note that Test Condition F should not be applied to Class S devices.

Table 2.5 can be used to establish the alternate time and temperature values. This table is based on the following two regression equations for Class B and Class S [9; 10], respectively:

For Class B:

$$T_b = 4.303 \times 10^{-4} e^{\frac{5{,}106.8}{273.15+T^*}}, \tag{2.1}$$

where

T_b = burn-in time is hours, and

T^* = burn-in temperature, °C.

For Class S:

$$T_b = 6.454 \times 10^{-4} e^{\frac{5{,}106.8}{273.15+T^*}}, \tag{2.2}$$

where

T_b = burn-in time in hours, and

T^* = burn-in temperature, °C.

Any time-temperature combination which is contained in Table 2.5 for the appropriate class may be used for the applicable test condition. The test conditions, duration and temperature, selected prior to test should be recorded and shall govern for the entire test.

2.5. Test Temperature

Unless otherwise specified, the ambient burn-in test temperature shall be 125°C minimum for conditions A through E (except for hybrids, see Table 2.5). At the supplier's option, the test temperature for Conditions A through E may be increases and the test temperature for Conditions A through E may be increased and the test time reduced according to Table 2.5. Since case and junction temperature will, under normal circumstances, be significantly higher than ambient temperature, the circuit employed should be so structured that the maximum rated junction temperature for test or operation shall not exceed 200°C for Class B or 175°C for Class S.

The specified test temperature is the minimum actual ambient temperature to which all devices in the working area of the chamber shall be exposed. This should be assured by making whatever adjustments are necessary in the chamber profile, loading, location of control or monitoring instruments, and the flow of air or other suitable gas or liquid chamber medium.

Table 2.4: Recommended burn-in times and temperatures for various test conditions

Minimum temperature T^*, °C	Minimum burn-in time, hr		Test Conditions	Minimum reburn-in time, hr
	Class S	Class B		
100	-	352	Hybrids only	24
105	-	300	Hybrids only	24
110	-	260	Hybrids only	24
115	-	220	Hybrids only	24
120	-	190	Hybrids only	24
125	240	160	A – E	24
130	208	138	A – E	21
135	180	120	A – E	18
140	160	105	A – E	16
145	140	92	A – E	14
150	120	80	A – E	12
175	-	48	F	12
200	-	28	F	12
225	-	16	F	12
250	-	12	F	12

2.6. Reliability after Burn-In

A factory test designed to catch systems with *marginal* components before they get out the door; the theory is that burn-in will protect customers by outwaiting the steepest part of the *bathtub curve*

Conditional reliability is useful in describing the reliability of a component or system following a burn-in period T_0 or after a warranty period T_0. We define conditional reliability as the reliability of a system given that it has operated for time T_0.:

$$R(t \mid T_0) = \Pr\{T > T_0 + t \mid T > T_0\}$$

$$= \frac{\Pr\{T > T_0 + t\}}{\Pr\{T > T_0\}} = \frac{R(T_0 + t)}{R(T_0)} \tag{2.3}$$

$$= \frac{\exp\left[-\int_0^{T_0+t} \lambda(t')dt'\right]}{\exp\left[-\int_0^{T_0+t} \lambda(t')dt'\right]} = \exp\left[-\int_{T_0}^{T_0+t} \lambda(t')dt'\right] \tag{2.4}$$

EXAMPLE 2.1

Let,

$$\lambda(t) = \frac{0.5}{1000}\left(\frac{t}{1000}\right)^{-0.5} \quad \text{Where, t is in years}$$

Which is a DFR. Then for reliability of 0.90

$$R(t) = \exp - \left(\frac{t}{1000}\right)^{1/2} = 0.90$$

And the design life is bound from

$$t = 1000(-\ln 0.90)^2 = 11.1 \text{ yr}$$

If we let $T_0 = 0.5$, a six month burn- in period, then

$$R(t \mid T_0) = \frac{R(t+0.5)}{R(0.5)}$$

$$= \frac{\exp-\left[(t+0.5)/1000\right]^{0.5}}{\exp-\left[(t+0.5)/1000\right]^{0.5}} = 0.90, \text{ and}$$

$$t = 1000\left[-In\,0.90 + \left(\frac{0.5}{1000}\right)^{0.5}\right]^2 - 0.5 = 15.8yr$$

This is an increase of over 4 years in the design life as a result of six- month burn-in period. This improvement in reliability from burn-in period T_0 will only be realized for a DFR as illustrated in the following example as shown in Appendix 2C.

$$\text{Let } \lambda(t) = \lambda t, \text{ an IFR for } \lambda > 0. \text{ Then}$$

$$R(t) = e^{-(1/2)\lambda r^2} \tag{2.5}$$

$$R(t \mid T_0) = \frac{e^{-(1/2)\lambda(t+T_0)^2}}{e^{-(1/2)\lambda T_0^2}}$$

Which can be simplified to:

$$R(t \mid T_0) = e^{-\lambda T_0 t} e^{-(1/2)\lambda r^2} \tag{2.6}$$

Since $\exp(-\lambda T_0 t)$ for $\lambda > 0$ is a decreasing function of T_0 increases.

For the reliability function given in Example 2.6, the conditional reliability is:

$$R(t \mid T_0) = \frac{1(t+T_0)^{2/a^2}}{1-T_0^{2/a^2}} = \frac{a^2-(t+T_0)^2}{a^2-T_0^2} \tag{2.7}$$

2.6.1. Residual MTTF

Since $R(t \mid T_0)$ is a reliability function, a residual MTTF may be obtained from

$$MTTF(T_0) = \int_0^{\alpha} R(t \mid T0)dt = \int_{T_0}^{\alpha} \frac{R(t')}{R(T_0)}dt' = \frac{1}{R(T_0)}\int_{T_0}^{\alpha} R(t')dt' \tag{2.8}$$

Where $t' = t + T_0$. For those units having survived to time T_0, Eq. (2.8) determines their mean remaining lifetime. For components having an IFR (DFR), one would expect the $MTTF(T_0)$ to be a decreasing (increasing) function of T_0, as shown in the following examples.

The reliability function $R(t) = (b-t)/b$ for $0 \le t \le b$ and zero elsewhere has an IFR. Its residual MTTF is given by

$$MTTF(T_0) = \left(\frac{b-T_0}{b}\right)^{-1} \int_{T_0}^{b} \frac{b-t'}{b}dt' = \frac{b}{b-T_0}\frac{(b-t')^2}{-2b} \Big|_{T_0}^{b} = \frac{(b-T_0)}{2} \text{ for } 0 \le T_0 \le b$$

The reliability function

$$R(t) = \frac{a^2}{(a+t)^2} = t \ge 0$$

Where $a > 0$ is a parameter (constant) of the distribution, has the hazard rate fuction

$$\lambda(t) = \frac{2}{a+t}$$

Which is decreasing. The MTTF is

$$MTTF(T_0) = \frac{(a+T_0)^2}{a^2}\int_{T_0}^{\alpha} \frac{a^2}{(a+t')^2}dt' = \frac{(a+T_0)^2}{a^2}dt' = \frac{(a+T_0)^2}{a^2}\frac{-a^2}{a+t'} \Big|_{T_0}^{\alpha} = a+T_0$$

This has the interesting property that the residual mean increases by the amount of the current age. If $T_0 = 0$, the unconditional mean, MTTF = a, is obtained.

2.7. A Physics of Failure Approach to IC Burn-In

Screening is a process that detects defects in a sub-population of devices that may exhibit early failure characteristics unlike the main population. Such defects occur due to multiple variabilities detected either through non-stress screens or by stress

screens, including burn-in. This section examines the problems with existing burn-in methods and presents a physics of –failure approach to burn- in.

2.7.1. Burn-In Philosophy

Burn -in has been used as a screen that subjects devices to extended operation at high temperatures to precipitate early failures and eliminate the infant mortality region. Traditionally, burn-in has been based on the bathtub curve failure pattern. The bathtub curve is used to determine the screening magnitude and level. However, these failure patterns have become outdated and as, a result, their relevance has diminished.

The goal to burn-in is to prevent failures from occurring in the field. Burn-in is typically a requirement imposed by the customer to demonstrate higher product reliability; manufacturers have different burn-in procedures for the same class components for military and commercial customers.

The typical burn-in procedure consists of placing parts in a thermal chamber for a specific amount of time under electrical bias. During and/ or after thermal environmental exposure, functional tests are conducted. Parts that fail the screen are discarded; parts that survive can be used.

2.7.2. Problem with Present Approach to Burn-In

A review of the burn-in practices used by some leading IC manufacturing reveals that even though burn-in has been regarded as a method for eliminating marginal devices with defects from manufacturing aberrations, the specifics of burn-in vary (Table 2.5). Most companies have their own burn-in specifications for commercial products; MIL-STD –833 is used to satisfy burn-in requirements for military products. Other companies use only MIL-STD- 833, but the selection of Method 1051 burn-in procedures for quality assurance also seems to be arbitrary. The emphasis is on empirical analysis, without any pointers to cost-effective application or subsequent manufacturing or assembly process modifications.

Burn-in present is a generic procedure consisting of a combination of time, steady-state temperature, and electrical stress. Burn-in procedures are often conducted without any prior identification of the nature of the defects to be precipitated, the failure mechanism active in the device, or their sensitivity to steady-

state temperature stress or without any quantitative evidence of the improvement achieved by the process.

By looking at the data from various companies burn-in has shown that it is ineffective for precipitating many failures. Data collected from various procedures sources shows that the majority of failures precipitated by burn-in are not valid. Valid failures include such things as mechanical damage, broken bond and broken package pins. Non-valid failures include such things as handling damage, for example, electrostatic discharge. It has been shown that burn-in detects less than 0.5% of failures of which less than 0.002% were valid(my burn-in article). Therefore, the failures burn-in precipitates are unlikely to occur in the field which defeats the purpose of burn-in

Burn –in may not precipitate many failures because it is performed under the widespread trust that the failure mechanism are steady-state temperature, temperature change, the rate of temperature change, or temperature gradients induce failures. An example that indicates that the failure are not steady – state dependent is presented. TriQuant Semiconductor found while testing their GaAs ICs that burn-in was ineffective in actuating any failure mechanisms. The reason were traced to dive architecture and failure mechanism that had no dependence on steady-state temperature.

Another reason could be because burn-in is conducted without prior knowledge of what failure mechanism is to be precipitated. The use of burn-in without attention to the dominant failure mechanism and the nature of their temperature dependencies is a misapplication of reliability concepts. Such use of burn-in may cause failure avoidance efforts, without yielding anticipated overall results, or expensive system implementations whose costs and complexities exceed the anticipated benefits in reliability.

Since burn-in does not precipitate many failures, some believe that it is more effective if it is applied for longer duration's. However, Motorola concluded, after numerous tests, that after 160 hours, the effectiveness of burn-in decreases significantly with the close to zero failures in the succeeding 1000 hours.. Other problems result due to burn-in which include.

- Palladium damage

- Increase in leakage currents
- Damage induced by additional handling

Burn-in has the potential to damage palladium lead finishes which can cause solderability problems. What happens is the finish on the leads disappears leaving a surface that cannot be soldered upon. It has also been shown that plastic parts degrade more severely than their ceramic counterparts after exposure to various radiations dosage levels(my article). The leakage current increases to the point where the performance is altered to an undesirable state. The reason for this is due to the presence of various materials present in the encapsulant which is not present in ceramic parts. During the burn-in process, parts are inserted and withdrawn from sockets, temperature chambers which makes them susceptible to additional handling damage. Handling damages that lead to failures include mechanical damage (e.g. bent leads), electrostatic discharge (ESD), and electrical overstress(EOS) failures. Many studies have been performed verifying the fact that burn-in is the source of EOS/ ESD damage.

Table 2.5: Burn-in time and temperatures

Source	Min. Temp. T_a (°C)	Time (hours)	Test condition	Comments
Mil-STD-833 Method 1015	100,105, 110,115, 120	Class:B 325,300, 260,220,190	Hybrids only	Either of the combinations of the cited temperature and time is used for burn-in of hybrids
	125	Class S: 240 Class B:160	A-E	Any of the specified combinations of temperature and time can be used for burn-in according to Method 1051 of MIL-STD-833. The various conditions of burn-in are defined by the electrical stress, steady state temperature (ranging from 100° C to 250° C) and time period (12 to 352 hours) Conditions include: • Test condition A: steady state temperature, reverse bias • Test condition B: steady state temperature, forward bias • Test condition C: steady state temperature, power and reverse bias • Test condition D: parallel excitation • Test condition E: ring excitation • Test condition F: temperature accelerated test [MIL-STD-883C, 1983: last revision incorporated 1990]
	130	Class S: 208 Class B:138	A-E	
	135	Class S: 180 Class B:120	A-E	
	140	Class S: 160 Class B:105	A-E	
	145	Class S: 140 Class B:92	A-E	
	150	Class S:120 Class B:80	A-E	
	175	Class B:48	A-E	
	200	Class B:28	A-E	

Source	Min. Temp. T_a (°C)	Time (hours)	Test condition	Comments
Mil-STD-833 Method 1015(cont)	225	Class:B 16	A-E	
	250	Class:B 12	A-E	
INTEL Corporation				
Intel Spec.	125°C	Memory products:48 hours	Method 1051 Condition C or D	Dynamic burn-in [Intel 1989, Intel 1990]
MIL-STD-833 Method 1051	125°C	Military products: 160 hours		

Advanced Micro Devices Inc. MIL-STD-833 Method 1051	125°C	Military products: 240 hours	Method 1051 Condition C or D	[Advance Micro Devices 1990]

Source	Min. Temp. T_a (°C)	Time (hours)	Test condition	Comments
LSI Logic Corporation MIL-STD-833 Method 1051	125°C	48 hours	Static or DC burn-in; dynamic burn-in	The results of production burn-in are measured as a percentage fallout rate or PDA (Percent Defective Allowable). The PDA calculation is simply the reject rate, the number of failures divided by the total number of devices in the lot, and the result compared against target PDA [LSI Logic 1990]

Source	Min. Temp. T_a (°C)	Time (hours)	Test condition	Comments
Texas Instruments Inc MIL-STD-833 Method 1051	125°C	MOS memory and LSI		The results of production burn-in are measured as a percentage fallout rate or PDA (Percent Defective Allowable). The PDA calculation is simply the reject rate, the number of failures divided by the total number of devices in the lot, and the result compared against target PDA [LSI Logic 1990]
MIL-STD-833 Method 1051		JAN S, monitored line, SEQ: 240 hours		
Power burn-in MIL-STD-750 Method 1039	25°C	Optocoupler screening: JAN, JANTX, JANTXV, 4N22,4N23,4N24JAN,4N22A,4N23A,4N24A:168 hours	V_{∞} = 20 Vdc V_{∞} = 10±5 Vdc PT= 275±25 mW I_F = 40 mA	

Components inserted into and extracted from sockets, temperature chambers, and pre-and post-test procedures can suffer additional handling damage in the form of bent leads and electrostatic discharge.

Historically, ionic contamination has been the dominant mechanism precipitated by burn-in. Sodium, potassium, or ions in the oxide of silicon MOS devices under bias and temperature lead to junction leakage and threshold voltage shifts that cause failure. Cool –down under bias and retest within 96hrs of burn-in

produces a relaxation of bias-induced charge separation. GaAs MESFET- based ICs have no oxide between the gate metalilization and the surface of the channel – the interface is a Schotkky diode. Similarly, the MESFET device unipolar, majority-carrier conducting, and reliant on the semi-insulating bulk material to achieve device isolation. There are no junctions and no leakage to consider.

In an effort to improve reliability, microelectronic manufactures have often subjected devices to increasingly longer periods of burn-in. However, Motorola noted that most of the failures precipitated by burn-in occur in the first 160 device hours, with few or no failures over the next 1,000 hours. This is controlled by the fact that the long-term projected failure rate, based on the number of failures over 1,000 hours, is the same order of magnitude as the actual measured failure rate over 1,000 hours.

2.7.3. A Physics- Of – Failure Approach to Burn –In

2.7.3.1. Understanding Steady –State Temperature Effects

The use of burn-in without attention to the dominant failure mechanisms and the nature of their temperature dependencies is a misapplication of reliability concepts. Such use of burn-in may cause failure avoidance efforts, without yielding anticipated overall results, or expensive system implementations whose costs and complexities exceed the anticipated benefits in reliability. While burn-in case be used for certain types of failures that can be accelerated by steady-state temperature effects, more insight into the failure mechanism can yield better solutions in terms of design and processes.

Microelectronic design and corrective action are often misdirected because of the confusion between quality and performance. Quality is a measure of the ability of a device to fulfill its intended function. Device performance, defined by electrical parameters such as threshold voltage, leakage currents, and propagation delay, is dependent on steady-state temperature, the device does not meet requirements. This may serve as an indicator for a design change, or for the unsuitability of the technology for high-temperature operation. Burn-in, where performance is checked after the temperature is lowered, will not uncover this type of problem.

2.7.3.2. Setting up the Burn-in Profile

A physics-of-failure approach to burn-in considers the potential material defects, design inconsistencies, and manufacturing variabilites for each process that could cause defects in the product. The burn-in methodology is an iterative process consisting of the following major steps.

- *Identify failure mechanisms, failure sites and failure stresses.*

Development of a burn-in program encompasses identifying potential failure mechanisms, failure sites, and failure stresses, active in a device technology. The burn-in process must be tailored to specific failure mechanism (s) at specific potential failure site(s) in order to be effective. The failure mechanism(s) and failure sites(s) depend on the materials and product processing technologies. Burn–in conditions are therefore specific to the manufacturing technology and hardware. The manufacturing sequence should be studied and possible defects, introduced due to the processing variabilites at each manufacturing stage, should be identified. The dominant failure stresses accelerating the failure mechanisms can be identified based on knowledge of the damage mechanics. The burn-in stress sequence will encompass those stresses that serve as dominant accelerators of the failure mechanisms.

- *Identify the combination of stresses to activate the identified failure mechanism cost effectively.*

Typically, there may be a number of failure mechanisms dominant in a device technology; each may have a dominant dependence on a different stress. Thus in order to activate all the failure mechanisms the dominant stresses need to be applied simultaneously and cost-effectively. To quantitatively determine the magnitude of stresses necessary to activate the failure mechanisms and arrive at the desired cost-effective combination of stresses, models must be developed for each failure mechanism, as a function of stresses, device geometry, material, and magnitude of defects and design inconsistencies. The quantitative models, however can aid only in relating the magnitude of a particular type of stress to manufacturing flaws and design inconsistencies, based on physics of failure concepts. The real case is more complex, involving interactions of stresses causing failure earlier than predicted by superposition of different stresses acting separately. There may be more than one failure mechanisms in a device technology. Each of the mechanisms will have its own dependence on steady state temperature, temperature cycle, temperature gradient,

and time dependent temperature change. An ideal case will be when an optimized combination of the relevant stresses is used to activate the failure mechanisms in a cost effective manner. The desired combination of stress is a function of the physics of the failure mechanism, and response of package material and configuration. The approach to arriving at the desired stress level consists of subjecting the components to discrete stress levels of steady state temperature, temperature cycle, temperature gradient, time dependent temperature change, and voltage. The selection of the temperature stress level should be based on the knowledge of designed for temperature of the device, the temperature of the device during operational life and the thresholds for various failure mechanisms. Conducting steps stress and HAST test for various magnitudes of each type of stress applied separately will give failure results in term of number of cycles to failure, failure mechanisms activated, and failure sites. From the test the stress levels required for activation of failure mechanisms will be identified.

- *Conduct burn-in and evaluate effectiveness.*

Burn-in should be assessed based on root cause failure analysis of the failed components, revealing the failure mechanisms, failure modes and failure sites. Inappropriate burn-in stresses will either damage good components by activating mechanisms not otherwise noticed in operational life, or allow defective parts to go through. To make sure the stress level and duration is right, the amount of damage (or the life consumed) for products without defects("good" products) must be evaluated. If necessary, the stress levels modified if necessary.

Product reliability (due to the design improvements) must be used as the index for subsequent burn-in decisions. Physics of failure approach is used to determine the effective acceleration of device dominant failure mechanisms and is given by: $A_{eff} = (A_T A_v A_x \ldots\ldots)$. Acceleration factors for dominant failure mechanisms are used to determine burn-in time (t_{bi}) and temperature (T_{bi}). The effective burn-in time is given by : $t_{eff} = A_T A_v A_x \ldots\ldots) t_{bi.}$

- *Decision regarding in-process monitors and burn-in modifications*

The above steps should be repeated until all products have the required expected life, with an optimized return on investment. The burn-in process should be augmented (supplemented or complemented) with in-line process controls to attain

the desired quality and reliability. The physics of failure models along with the burn-in results will determine the optimal manufacturing stress levels and process parameters for minimal defects level. The in-line process controls will ensure that the process parameters are maintained at their optimal values to minimize the occurrence of defects.

Economic analysis should indicate whether the burn-in should be continued or modified. A cost-effective burn-in program addresses all the relevant failure mechanisms by employing a minimum set of devices. Burn –in is recommended for all products which are in the development stage and do not have a mature manufacturing process. Burn-in at this stage not only improves the quality and reliability of the products but also assists in determining product and process (manufacturing, assembly, testing) corrective actions. Products with a standardized design and a relatively mature process need burn-in only if the field failure returns indicate infant mortality. A cost analysis and return on investment is conducted to calculate the economics of the burn-in program. Analysis of cost and return of investment based on the customer satisfaction and the hidden factory costs(the costs associated with the factory inputs which do not add value to the product, like product inspection, testing, rework, etc.) determine the profits to an organization. Burn in economics are critical in convincing management about the benefits that accrue from burn-in and provide a benchmark for making improvements in the next burn-in "cycle".

3. Reliability Evaluation Using Databooks

This chapter presents two papers which discuss the problems associated with reliability evaluation using MIL-HDBK-217 or any data book. Second paper presents a new methodology of system reliability evaluation called as CRAM (Consolidated Reliability Assessment Method).

3.1. MIL-HDBK-217 vs. HALT/HASS

For the last three decades, MIL-HDBK-217 has been widely used to predict product reliability. Today, however, highly accelerated life testing (HALT) and highly accelerated stress screening (HASS) are being recognized as effective tools to intensify product reliability.[2] The military standard and HALT/HASS cover different areas in the reliability world. Is there any correlation between them?

Manufacturers usually make reliability predictions based on failure models described in MIL-HDBK-217, Bellcore TR-332, or some other model before the product is manufactured or marketed.[3,4] But when a product is delivered to customers and then field failure reports begin to arrive, the preliminary reliability prediction sometimes is not validated by real-world failure reports.

Some manufacturers have said the prediction model could be widely inaccurate when compared with the performance in the field. What makes the discrepancy between the reliability prediction and the field failure report?

3.1.1. The Purpose of MIL-HDBK-217

This military standard is used to estimate the inherent reliability of electronic equipment and systems, based on component failure data. It consists of two basic prediction methods:

- *Parts-Count Analysis:* Requires relatively little information about the system and primarily uses the number of parts in each category with consideration of part quality and environments encountered. Generally, the method is applied in the early design phase, where the detailed circuit design is unknown, to obtain a preliminary estimate of system reliability.

- *Part-Stress Prediction:* Uses complex models composed of detailed stress-analysis information as well as environment, quality applications, maximum ratings, complexity, temperature, construction, and a number of other application-related factors. This method tends to be used near the end of the design cycle, after the actual circuit design has been defined.

The general failure mod-el in MIL-HDBK-217 and Bellcore TR-332 is of the form:

$$\lambda_p = \lambda_b \pi_Q \pi_E \pi_A \bullet\bullet\bullet$$

Where, λ_b = the base failure rate, described by the Arrhenius equation

$\pi_Q\pi_E\pi_A,\ldots$ = factors related to component quality, environment, and application stress

The Arrhenius equation illustrates the relationship between failure rate and temperature for components. It derives from the observed dependence of chemical reaction, gaseous diffusion, and migration rates on temperature changes:

$$\lambda_b = K @ \frac{-E}{\kappa T}$$

where: λ_b = process rate (component failure rate)

E = activation energy for the process

κ = Boltzmann's constant

T = absolute temperature

K = a constant

Detailed models are provided for each part type, such as microcircuits, transistors, resistors, and connectors.

3.1.2. The Merit of HALT/HASS

HALT is performed during design to find the weak reliability links in the product. The applied stresses to the product are well beyond normal shipping, storage, and application conditions. HALT consists of:

- Applying environmental stress in steps until the product fails.
- Making a temporary change to fix the failure.
- Stepping stress further until the product fails again, then fix it.

- Repeating the stress-fail-fix process.
- Finding fundamental operational and destruct limits of the product.

HASS is performed in the production stage to confirm that all reliability improvements made in HALT are maintained. It ensures that no defects are introduced due to variations in the manufacturing process and vendor parts. It contains the following:

- Precipitation screen for latent defects to be turned into patent defects.
- Detection screen to find patent defects.
- Failure analysis.
- Corrective actions.

The precipitation and detection screen limits of HASS are based on HALT results. Usually, the precipitation-screen limits are located between operational limits and destruct limits and the detection screen limits between spec limits and operational limits, as shown in Figure 3.1.

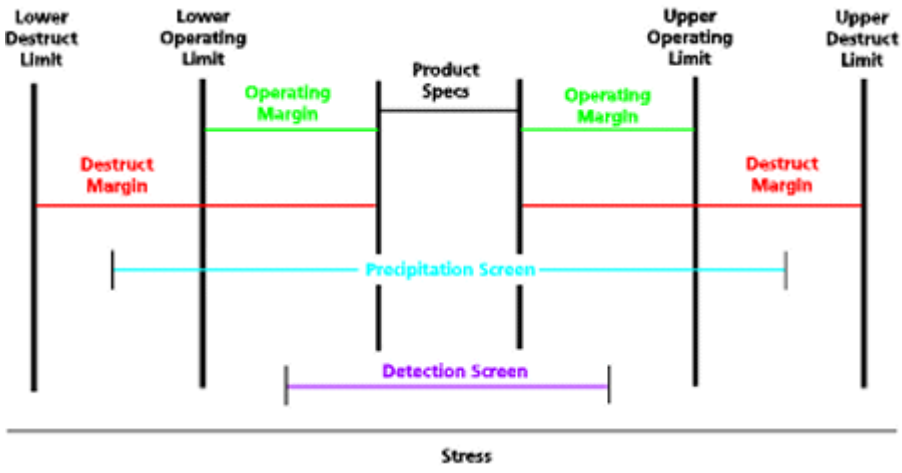


Figure 3.1: Hass Limits Selected From HALT Data

HALT/HASS has been proven to find latent defects that would very likely precipitate in end-use applications, causing product failures in the field. As a result, the HALT/HASS process can effectively intensify product reliability.

3.1.3. Why MIL-HDBK-217 Turns Out Inaccurate Predictions

The prediction techniques described in MIL-HDBK-217 for estimating system reliability are based on the Arrhenius equation, an exponentially temperature-dependent expression. But many failure modes in the real world do not follow the equation.

For instance, mechanical vibration and shock, humidity, power on/off cycling, ESD, and dielectric breakdown—all independent of temperature—are common causes of failure. Even some temperature-related stresses, such as temperature cycling and thermal shock, would cause failures that do not follow the Arrhenius equation.

More importantly, the reliability of components in many electronic systems is improving. Consequently, component failure no longer constitutes a major reason for system failure. But, the MIL-HDBK-217 model still tells us how to predict system reliability based on part failure data.

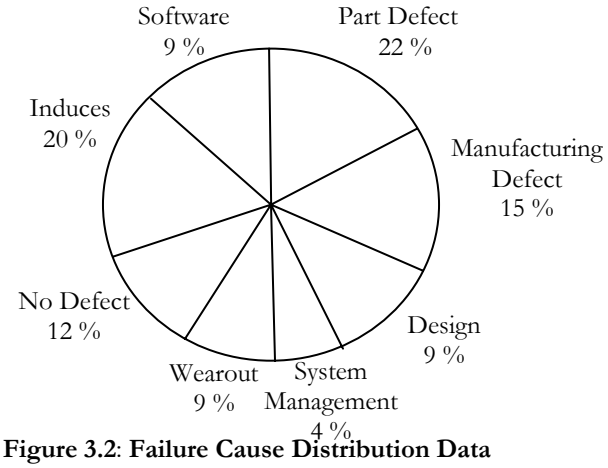


Figure 3.2: **Failure Cause Distribution Data**

Figure 3.2 illustrates the nominal percentage of failures attributable to each of eight predominant failure causes, based on data collected by the Reliability Analysis Center.[6] The definitions of the eight failure causes in Figure 3.2 are as follows:

- Parts—22%: Part failing to perform its intended function.
- Design—9%: Inadequate design.
- Manufacturing—15%: Anomalies in the manufacturing process.
- System Management—4%: Failure to interpret system requirements.

- Wear-Out—9%: Wear-out-related failure mechanisms.
- No Defect—20%: Perceived failure that cannot be reproduced upon further testing. These failures may or may not be actual failures; however, they are removals and count toward the logistic failure rate.
- Induced—12%: An externally applied stress.
- Software—9%: Failure to perform its intended function due to a software fault.

To illustrate the disparity, consider the following: A circuit board containing 338 components with six component types is used in a mobile radio system.[4] The failure rate of the MIL-HDBK-217 prediction is 1.934 failures per million hours, as shown in Table 3.1. The field behavior of the board, however, shows 19 failures in a total operating time of 4,444,696 hours, resulting in a field failure rate of 4.274 failures per million hours. The deviation 4.274 - 1.934 = 2.34 failures per million hours was not covered by the MIL-HDBK-217 prediction.

Table 3.1: Contribution to Failure Rate of Each Component in MIL-HDBK-217 Prediction

Component	Ceramic Capacitor	Diode	Bipolar IC	Resistor	Bipolar Transistor	Tantalum Capacitor	Failure Rate
Calculated Failures	0.004	0.009	0.05	0.052	1.225	0.594	1.934

Actually, many field failures are caused by unpredictable factors, often the main reasons for reliability problems in today's electronic systems. But those unpredictable reasons can be successfully precipitated, detected, and eliminated during a HALT/HASS process.

3.1.4. Conclusion

Before making a reliability prediction, be certain of one of the two following items:

1. The failure modes described in the prediction model account for the vast majority of system failures. If not, go to b.
2. Prediction is made after reducing unpredictable defects by performing HALT/HASS.

3.1.5. References

1. MIL-HDBK-217, Reliability Prediction of Electronic Equipment, U.S. Department of Defense.
2. Hobbs, G., Accelerated Reliability Engineering HALT and HASS, John Wiley & Sons, 2000.
3. Bellcore TR-332, Issue 6, Reliability Prediction Procedure for Electronic Equipment, Telcordia Technologies.
4. Jones, J. and Hayes, J., "A Comparison of Electronic-Reliability Prediction Models," IEEE Transactions on Reliability, Vol. 48, No. 2, June 1999, pp. 127-134.
5. Leonard, C. T. and Pecht, M., "How Failure Prediction Methodology Affects Electronic Equipment Design," Quality and Reliability Engineering International, Vol. 6, 1990, pp. 243-249.
6. Denson, W., "A Tutorial: PRISM," RAC, 3Q 1999, pp. 1-2.

3.2. A New System-Reliability Assessment Methodology

3.2.1. Abstract

This paper outlines the structure for a new electronic system reliability assessment methodology. The term "system" is used because the methodology accounts for all predominant causes of system failure. This approach goes beyond traditional approaches such as MIL-HDBK-217 or Belcore. These prediction practices focus on the inherent capabilities and limitations of device technologies. The new Consolidated Reliability Assessment Model (CRAM) extends the reliability modeling to include special cause failure drivers such as those due to "design defects." System level failure drivers, such as "requirements deficiencies" are also measured as to their effect on reliability. Our studies show these added modeling factors most significant failure drivers today.

The new model adopts a broader scope to predicting reliability it factors in all available reliability data as it becomes available on the program. It thus integrates test and analysis data, which provides a better prediction foundation and a means of estimating variances from different reliability measures.

There is much information available in the design and development of modern electronic systems that can potentially be beneficial in providing data and information useful for quantifying system reliability. Example of such information includes analysis performed early in the design phases (reliability prediction, FMEA, thermal analysis, etc.), process information (design, component selection, manufacturing) and test data (reliability qualification, demonstration, life test, performance testing, etc.). The CRAM captures and integrates the best information from all sources to produce a consolidated reliability estimate of the product. Particular goals of this model are to:

- Estimate system failure rate and its variance
- Explicitly recognize and account for special (assignable) cause problems
- Model reliability from the user (or total system level) perspective
- Provide and intuitive reliability model structure
- Promote cross-organizational commitment to Reliability, Availability and Maintainability (RAM)

- Quantitatively grade developers' efforts to affect improved reliability
- Maintain continuing organizational focus on RAM throughout the development cycle
- Integrate all RAM data that is available at the point in time when the estimate is performed (analogous to the statistical process called imeta-analysis)
- Provide flexibility for the user to customize the reliability model with historical data
- Impact (positively) the product development process and the resulting developed product

3.2.2. Background

3.2.2.1. Need for the Model

Recent advances in government and industry have set the pace for development of new reliability assessment methods. In 1994, Military Specifications and Standards Reform (MSSR) was initiated. MSSR decreed the adoption of performance based specifications as a means of acquiring and modifying weapon systems. It also overhauled the military standardization process. MSSR led to the creation of the "105 Heartburn Specifications & Standards & List" a list of standardization documents that required priority action because they were identified as barriers to commercial processes, as well as major cost drivers in defense acquisitions. The list included only one handbook, MIL-HDBK-217, "Reliability Prediction of Electronic Equipment." Over the years, several vocal critics of MIL-HDBK-217 have complained about its utility as an effective method for assessing reliability. While the faultfinders claim that MIL-HDBK-217 is inaccurate and costly, to date no viable replacement methods are available in the public domain. As the DoD Lead Standardization Activity for reliability and maintainability (R&M), Rome Laboratory (RL) was responsible for implementing the R&M segment of MSSR. With this, RL initiated a project to develop a new reliability assessment technique to supplement MIL-HDBK-217, and to overcome some of its perceived problems.

Utilizing standardization reform funding, RL awarded a contract to a team composed of personnel from IIT Research Institute (IITRI) and Performance Technology. The objective of the contract, the results to date which are summarized

in this paper, was to develop new and innovative reliability assessment methods. The contract called for the development of models flexible enough to suit the needs of system reliability analysis regardless of their preferred (or required) initial prediction methods. The intent was to use the final product to supplement or possibly replace. MIL-HDBK-217. The contract was broken down into three phases. Phase I identified and analyzed all existing initial reliability assessment methodologies including empirical methods and physics-of-failure based models, generic system and component level models, similar system data model, and test data models. The purpose and effectiveness of the various methods were studied. Phase II derived methodologies for improving the accuracy of reliability predictions and assessments with information obtained from the design, manufacturing, part selection test, and software development processes. Phase III involved development and automation of the new reliability assessment model.

3.2.2.2. Uses for Reliability Predictions

In an effort to identify the manner in which reliability predictions are used by reliability practitioners, a survey was issued, for which approximately sixty non-DoD companies responded. From this data, the predominant purpose for performing reliability assessment, in order of frequency are:

1. Determining feasibility in achieving a reliability goal or requirement
2. Aiding in achieving a reliable design (i.e., derating component selection, environmental precautions, input to FMEAs/Fault Trees)
3. Predicting warranty costs and maintenance support requirements

3.2.2.3. Methodologies Used In Performing Predictions

Survey respondents were also asked to identify the methodologies they use when predictions are performed. MIL-HDBK- 217 was determined to be the most universally applied failure rate prediction methodology. Several companies have adapted it by adding detailed manufactureris data or test data when it is available. Those who have evolutionary products have been able to successfully tailor their predictions based upon field experience with predecessor products. This adaptation has often been accomplished by making suitable modification to the quality pi factor. Most respondents have stated that they would like a methodology that is more

reflective of state-of-the-art technology. Some indicate that the constant failure rate is not truly representative. Respondents also indicated that they would also like one that addresses the special cause or process concerns, and a tool that proactively aids in the development of a reliable design.

The CRAM model, like all models, is a mathematical representation of a physical situation. The degree to which a model is successful depends on its technical merit and on its acceptance be reliability engineering practitioners. The technical validity of the model depends on many factors including the assumptions made in model development and the data on which the model is based (and its associated limitations). The success of the model, in terms of acceptance, depends on the degree to which the model is intuitive, and the sensitivity of the model to parameters that are of interest to the model user.

While some in the reliability profession believe that reliability modeling should focus on the development of models that attempt to deterministically model failure causes, it is well established that system reliability failure causes are not driven by deterministic processes, but rather they are stochastic processes and must be treated as such in a successful model. This does not simply that known failure causes should not be studied such that design and processing changes can be implemented to preclude their occurrence. Striving for this is always good reliability engineering practice. However, or the purposes of quantifying he expected field reliability of a system, a deterministic approach is not practical. For example, there are many more factors that influence reliability than can be accounted for in a model. It is because of this that there is a similarity between reliability prediction and chaotic processes. This likeness stems from the fact that the reliability is entirely dependent upon initial conditions (e.g., manufacturing variation) and use variables (i.e., field application). Both the initial conditions and the application variables are often unknowable. For example, the likelihood of a specific system containing a defect is often unknown, depending on the defect type, because the propensity for defects is a function of many variables and deterministically modeling them all is clearly impossible. Additionally, the specific stresses to which the system will be exposed during its lifetime cannot be ascertained and quantified with any significant degree of confidence.

As a result, a successful model will realistically estimate system reliability as a function of the known entities with quantifiable confidence in the estimate. The CRAM model strives to generally describe the reliability behavior of systems by estimating the effects of known failure drivers on system reliability, and by doing so in a fashion intuitive to reliability engineers.

3.2.3. The Basis for a New Model

3.2.3.1. Uncertainty in Traditional Approach Estimates

A goal of the CRAM model is to model predominant system reliability drivers. The premise of traditional methods, such as MIL-HDBK- 217, is that the failure rate is primarily determined by the technology and application stress of the component comprising the system. This was a good premise when components exhibited higher failure rates and systems were not as complex as they are today. Increased system complexity and component quality have resulted in a shift of system failure causes away from components to more system level factors including system requirements, interface and software problems. A significant number of failures also stem from non-component causes such as defects in design and manufacturing. Hisorically, these factors have not been explicitly addressed in prediction methods. The approach used to develop the CRAM model was to; 1) Quantify the uncertainty in predictions using the "component based" traditional approaches and 2) Explicitly model the factors contributing to that uncertainty. Table 3.2 presents the multipliers of the failure rate point estimate as a function of confidence level. This data was obtained by analyzing data on system for which both predicted and observed data was available. For example, using traditional approaches, one could be 90 % certain that the true failure rate was less than 7.575 times the predicted value.

Table 3.2: Uncertainty Level Multiplier

Percentile	Multiplier
.1	.132
.2	.265
.3	.437
.4	.670
.5	1
.6	1.492
.7	2.290
.8	3.780
.9	7.575

3.2.3.2. System Failure Causes

Predominant causes of system failure were identified and their relative probability of occurrence baselined for modern electronic systems. A summary of the data used to accomplish this was collected from a survey and is illustrated in Figure 3.2. Each cause can be further broken down into their constituent causes. For example, parts can be further apportioned amongst part defect, induced and wear-out.

The above pi chart values represent the average percentages attributable to each failure cause. Also analyzed was the variance around these percentages. Figure 3.3 illustrates the 30th, and 70th percentiles for each of the four categories comprising the intrinsic failure rate. These Pi factors are unit less failure rate multipliers as a function of the grade (in percentile) of each cause. For example, the Pi factor corresponding to a 30th percentile grade is approximately 1.1, whereas he factor corresponding to the 50th percentile is approximately 0.45

It is these distributions around the mean percentage values that form the range within which the failure rate estimate is scaled, and the failure rate estimate for each failure cause is determined. The conclusion that can be made based on these observations is that parts, while still a significant reliability factor, do not contribute to system reliability to the extent implied by traditional estimation.

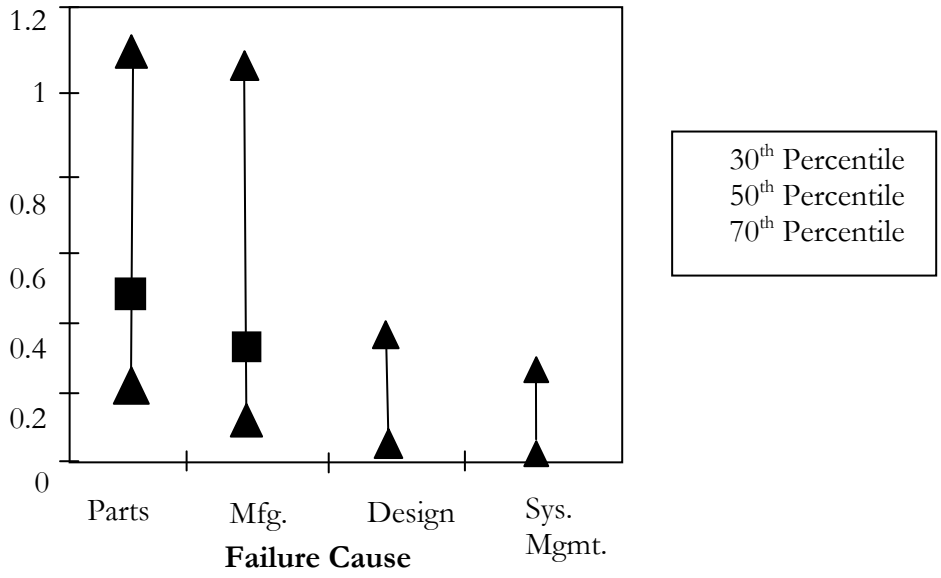


Figure 3.3: **Pi Factor for the 30th, 50th and 70th Percentile Grade**

3.2.3.3. Model Description

The achievement of system reliability relies on the following elements:

1) Obtaining valid system requirements and managing interface dependencies
2) Good quality parts must be chosen
3) The parts must be designed into a system in a robust manner to insure that they can meet performance requirements over the systems design life
4) Design must be validated through analysis and test
5) The system must be manufactured without inducing damage or adding defects
6) A management philosophy that supports the achievement of the above three elements

The adequacy of a company in achieving the above four factors will dictate the degree to which a reliable end-item is achieved. Therefore, a methodology that quantifies system reliability must account for the degree to which a company implements the processes required to mitigate the probability of failure due to the above categories. The basis of the methodology developed in this study is that the adequacy of this implementation can be graded and a percentile rating can be obtained. This percentile corresponds to the percentage of all companies (in a given industry) that have processes in place (for each failure cause) that are worse than the company being rated.

The predictive modeling takes place in several successive stages. First, an initial reliability prediction is performed to derive a "seed" failure rate estimate. This can be accomplished using any viable technique. Then, the Development Grading Process Model is used. This model essentially grades the development effort with its likely affect on the mitigation of special cause problems. The process grading factors are first judged in the program planning stages. The actual factors are then updated according to real practice. These updates are done in accordance with the timing for updating the reliability predictions. Next, the initial prediction is combined with the Process Grades to form the best pre-build failure rate estimate. The flow of the model developed in this study is given in Figure 3.4.

Combining the initial prediction with process grades consists of adjusting the failure rate in accordance with the level to which processes have been implemented

that mitigate the risk of failures associated with each failure cause. In similar assessment for software is undertaken; and the failure rates for hardware and software are added. The CRAM (Consolidated Reliability Assessment Methodology) block in Figure 3.4 refers to the methodology of mathematically combining the initial assessment, process grades, operational profile and software assessments.

The mathematical model form for this inherent failure rate is:

$$\lambda_P = \lambda_{IA}\left(\prod_P + \prod_D + \prod_M + \prod_S\right) + \lambda_{SW} + \lambda_W \tag{3.1}$$

The logistics failure rate accounts for factors attributable to the induced and no defect found categories, and is:

$$\lambda_P = \lambda_{IA}\left(\prod_P + \prod_D + \prod_M + \prod_S + \prod_I + \prod_N\right) + \lambda_{SW} + \lambda_W \tag{3.2}$$

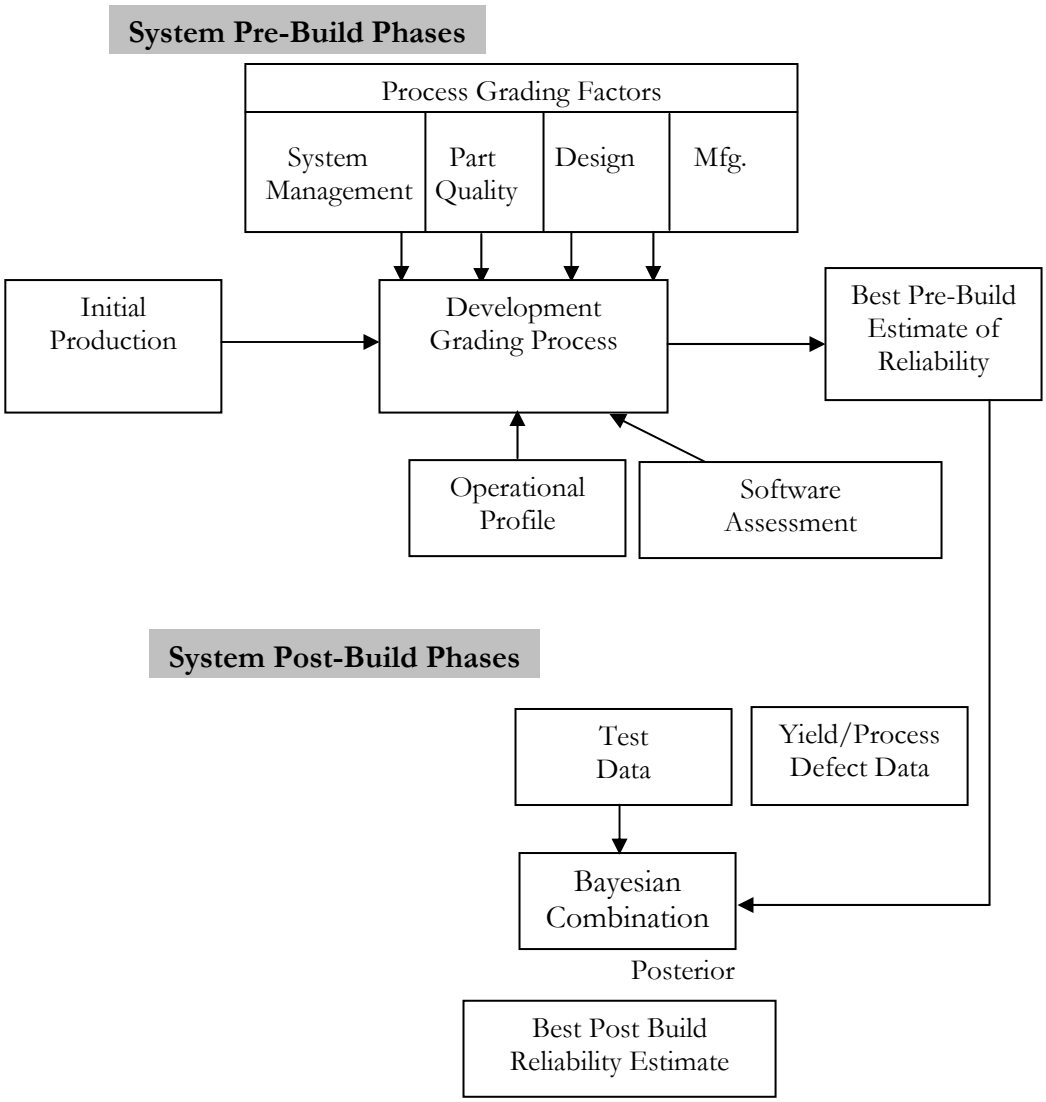


Figure 3.4: **Reliability Assessment Modeling Approach**

Where:

λ_P = Prediction system failure rate

λ_{IA} = Initial multiplier, function of parts process grade

$\prod_P$ = Part multiplier, function of parts process grade

$\prod_D$ = Design multiplier, function of design process grade

$\prod_M$ = Manufacturing multiplier function of manufacturing process grade

$\prod_S$ = System management multiplier function of management process grade

$\prod_I$ = Induced multiplier, function of design precautions taken to mitigate induced failures

$\prod_N$ = No Defect Found (NDF) multiplier, function of NDF process grade

$\prod_{SW}$ = Software failure rate

$\prod_W$ = Failure rate associated with wearout failure modes

At this point in the System Assessment Methodology, the best "pre-build" estimate of failure rate is obtained. The next step is to combine this "pre-build" estimate with any empirical data that is available. Once this occurs, the best system failure rate estimate is obtained.

The basic premises on which the model presented in this paper is based are as follows:

1. Much of the variability in actual reliability relative to the predicted reliability based on traditional methods is a result of variations in the processes that are used to design and build the system.
2. The causes of system failure stem from mutually exclusive primary categories. These categories are Parts, Design, Manufacturing, System Management, and Software, induced, and No defect found.
3. The traditional approaches to system reliability prediction (i.e., those that predict reliability as a function only of the component comprising the system) implicitly include failure rates attributable to non-component failure causes, such as design deficiencies, manufacturing errors, etc. This model explicitly measures these effects.

4. The traditional approaches to system reliability prediction, with modification based upon the user's particular field experience, can estimate a reasonably accurate system failure rate.

Figure 3.3 is a more detailed flow diagram representing the assessment model.

The following sections provide more detail on the various elements of the methodology.

3.2.3.4. Initial Assessment

The methodology starts with an initial reliability assessment. This assessment is intended to be the baseline reliability estimate for the system being analyzed. It will be enhanced as process grading factors and empirical data are incorporated in the assessment. There are several options for this assessment. First, if an analysis has already been performed on the system, then it should be used for this purpose. If one has not been performed, estimates can be made using generic system level prediction techniques.

3.2.3.5. Process Grading

An objective of the CRAM model is to explicitly account for the factors contributing to the variability in traditional reliability prediction approaches. This accomplished by grading the process for each of the failure cause categories. The resulting grade for each cause corresponds to the level to which an organization has taken the action necessary to mitigate the occurrence of failures of the cause.

The sum of the $\prod$ factors within the parenthesis in Equation 2 is equal to unity for the average grade. For example, the nominal percentage of failures due to parts is 32 %. Therefore, the $\prod_P$ is equal to .32 if an average process grade (50th percentile) is obtained. Likewise, it will increase if "less than average" processes are in place and decrease if better than average processes are in place.

Space does not permit presentation of the process grading factors for all failure causes, but for illustration purposes, a summary of each major category of grading factor is presented in Table 3.3.

Each of the categories in Table 3.3 are further broken down into its constituent elements. For example, the specific elements comprising the engineering skill category of design are given below.

- Is the development program organized as "cross functional development teams" (CFDT) involving: design, manufacturing, test, procurement, etc.?
- % of team members having relevant process experience, i.e., they have previously developed a product under the current development process
- % of engineering team that are degreed
- % of engineering team having advanced technical degrees
- % of engineering team having advanced technical degrees
- of engineering team members, in the past year, having patents, papers, professional registration, or held professional society offices
- % of engineering team members who have taken engineering courses in the past year
- Are resource people identified, for program technology support, in key technology and specialty areas such as ada design, opto-electronics, servo control, ASIC design, etc. To provide program guidance and support as needed?
- Are resources people identified, for program tools support, to provide guidance and assistance with CAD, simulation, etc.?
- % of design engineering people with cross training experience in manufacturing or field operations

3.2.3.6. Adding Software Failure Rate

Software can be the dominant failure contributor in some complex systems (Ref. 4). Due to its importance as a reliability driver, a separate additive failure rate model has been developed. System test is the development point where reliability growth testing of software beings, at which point the total system is finally integrated and simulated in the intended application. Failures occur, and the underlying faults found are isolated and removed. The MTBF of the software improves as the faults are fixed.

This is where software reliability is traditionally measured (Ref. 7). The model developed here allows one to predict reliability of the software before code is written. The only inputs required are the estimated extent of code and the process maturity level under which it was developed.

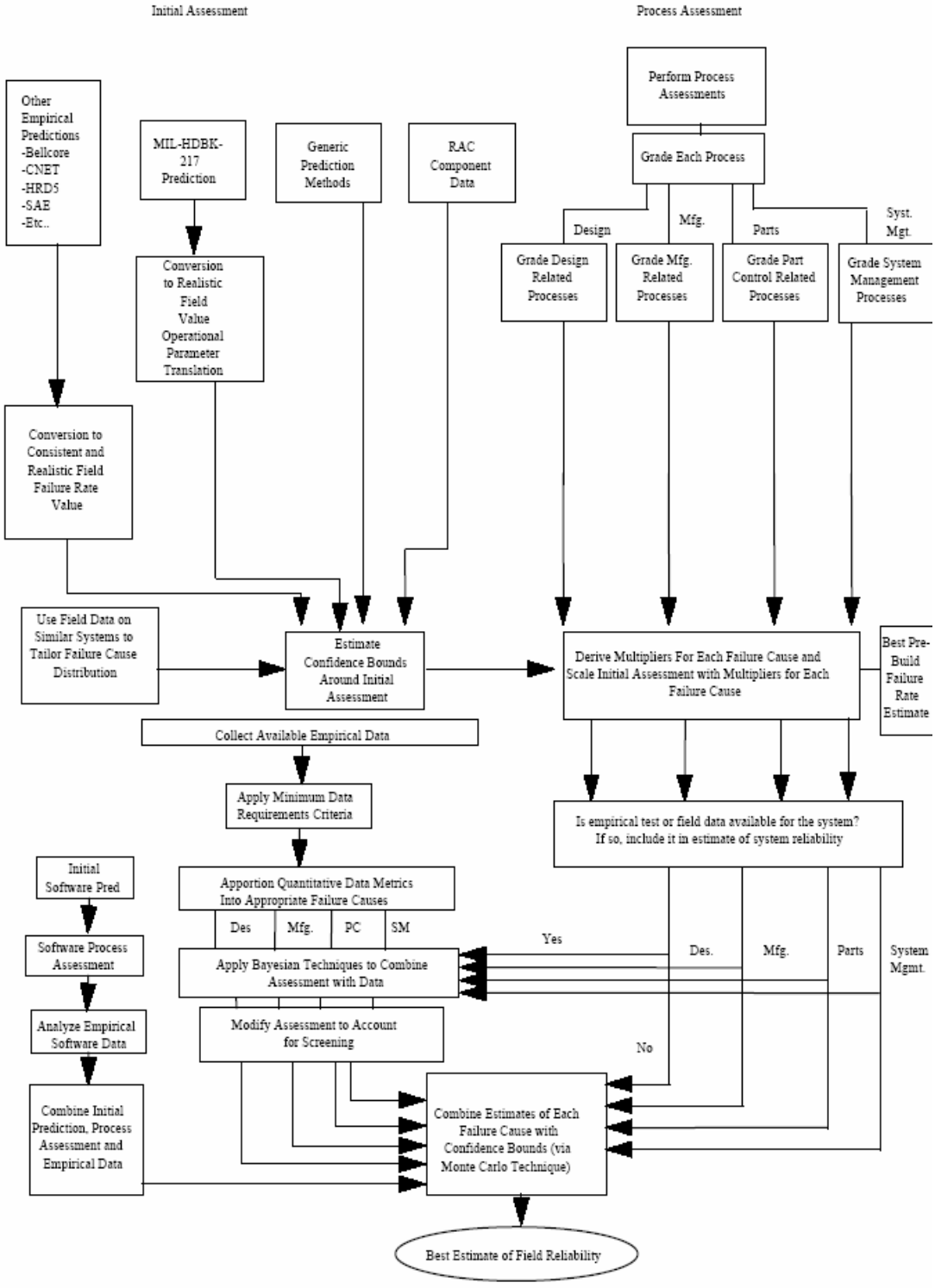


Figure 3.5: Assessment Model Flow

Table 3.3: Process Grading Categories

Design	• Engineering Skill • Technology Maturity • Design Margin and Optimization • Design for Manufacturability • Development Process • Development Process Metrics • Development Documentation • System Complexity • Product Testing • Engineering Design Automation Tools
Manufacturing	• Design for Manufacturability • Personnel Characteristics • Data and Analysis • Manufacturing Leadership • Manufacturing Process • Testability • Technology • Solder Joint Spacing
Parts Management	• Part Selection Criteria • Parts Suppliers Criteria • Custom Part Support • Critical Parts Support
System Management Considerations	• Requirements Management • Requirements Extensibility • Product Vision • Requirements Traceability and Verification • Interface Management • Operation Management • Change Management • Risk Management
Could Not Duplicate or No Defect Found	• Design Considerations • Analysis and Process Considerations

In this methodology, the latent fault density (design measure) is mapped to a probable field Mean Time Between Failure (MTBF), which is a user measure. This modeling is developed from a largely empirical basis. It establishes a basis to perform early predictions of software reliability. Two software development parameters are typically available in the planning stages. These development parameters form the front end of this model:

(1) The Estimated Code Size. This estimate is generally expressed in thousands of lines of (executable) source code or KSLOC. Commentary code is excluded from the code sizing.

(2) The Quality Level Of The Development Process. This is most commonly measured in terms of the Software Engineering Institute (SEI) process

capability level measure. SEI rates software providersi process capability from a beginning Level of 1 to the highest Level of 5.

These ratings correspond to Initial process capability (Level 1) up to optimizing process capability (Level 5). There are two other measures used to rank the development process. One is the ISO 9000 quality certification. It is a two-level measure. Either a company is ISO 9000 certified or it is not certified. The other is he development process rating measures of the air traffic system, the RTCA safety levels. The development correlation to fault density notion is further extended here to include the ISO 9000 ratings and the air traffic safety ratings. The latter are under Requirements and Technical Concepts for Aviationi (1992) RTCA safety levels (A, B, C, D, and E in decreasing levels of stringency). The predictive capability of all of these measures can be improved by adjusting the projected fault density by relevant experience data that the development organization has on the fielded code of its predecessor products.

The assertion in this model is that the software process capability is a good predictor of the latent fault content shipped with the code. The better the process, the better the process capability ratings and the better the code developed under that process.

3.2.3.7. Adding Failure Rate Due to Wearout Modes

If a physics of failure analysis is performed, its value is that it quantifies the reliability of specific wearout related failure mechanisms. It does not however, quantify the reliability of components or assemblies as a function of manufacturing defects, design inadequacies or induced failures. The assessment methodology accounts for this by adding a failure rate associated with wearout failure mechanisms. This failure rate is quantified based on the physics of failure analysis.

3.2.3.8. Logistic Failure Rate Contributions

The logistics failure rate refers to the replacement are in the field application of a system. Equation 1 addresses the prediction of inherent reliability of a system. Of equal importance to some reliability practitioners is the prediction of a system maintenance rate. This is addressed by Equation 2. Whether an analyst should be predicting the inherent failure rate of a system or its maintenance rate depends upon his or her particular goals and how the prediction is to be used.

It is desirable to explicitly model the factors affecting both failure rate and maintenance rate. The maintenance failure rate, or logistics failure rate, includes the inherent failure rate plus induced failures (I), and No Defect Found (NDF) failure contributions. The logistics failures are a super set of failure rate. The induced and No Defect Found category processes are graded in a similar manner as the Part, Manufacturing, Design and System Management categories.

3.2.3.9. Adding Empirical Data

The user of this model is encouraged to collect as much empirical data as possible, and use it in the assessment. This is done by mathematically combining the assessment made (base on the initial assessment and the process grades) with empirical data. This step will combine the best ipre-buildi failure rate estimate obtained from the initial assessment (with process grading) with the metrics obtained from the empirical data. Bayesian techniques are used for this purpose. This technique accounts for the quantity of data by weighting large amounts of data more heavily than small quantities.

The manner in which this is accomplished is to apply the following equation:

$$\lambda = \frac{a_o + a_1 + \ldots a_n}{b_o + b_1 + \ldots b_n}$$

Where:

λ = The best estimate of the predicted failure rate

a_o = The equivalent number of failures of the prior distribution corresponding to the reliability prediction (after process grading has been accounted for)

b_o = The equivalent number of hours associated with the reliability prediction (after process grading)

a_1 through a_n = The number of failures experienced in the empirical data. There may be n different types of data available

b_1 through b_n = The equivalent number of cumulative operating hours (in millions) experienced in the empirical data. These values must be converted

to equivalent hours by accounting for the accelerating effects between the test and use conditions.

3.2.4. Future Plans

A logical next step is to transition the new technique to potential users. In the near future, a useable version of the new CRAM model will be available in several mediums. A computerized version is near completion and will be available from Rome Laboratory. The next revision of MIL-HDBK-217 may include the model as an appendix. The Institute of Electrical and Electronics Engineers (IEEE) is considering the development of a new standard, guide, or recommended practice based on the CRAM. The American Society for Quality Controls Reliability Division plans to continue publishing updates on the new model in the Reliability Review Journal. The Reliability Analysis Center (RAC) also has plans to include CRAM in an upcoming RAC document. The RAC also plans to develop component prediction models that can be sued to perform more accurate initial assessments than those models currently available.

An important aspect of technique development which is often overlooked is the process of peer review. Now that Version I of the CRAM model is complete, the developers are seeking feedback and suggestions for improvement. If you have any comments or recommendations, please write to Rome Laboratory/ERSR, Attn.: Joe Caroli, 525 Brooks Rd, Rome NY 13441-4505, email: carolij@rl.af.mil.

3.2.5. References

1. Denson, W.K. and S.K Kenne, "New System Reliability Assessment Methods", RAC Project A06839, March 17, 1997.
2. Chillarege, Ram; Biyani, Shriram; and Rosenthal, Jeanette, iMeasurement of Failure Rate in Widely Distributed Software, The 25th Annual Symposium on Fault Tolerant Computing, IEEE Computer Society, June 1995.
3. Cole, G.F., and Keene, S.J., iReliability Growth of Fielded Softwarei, ASQC Reliability Review, Vol. 14, March 1994.
4. Koss, E.W., iSoftware Reliability Metrics for Military Systems, I Proc. Reliability and Maintainability Symposium, 1988, Los Angeles, California.
5. Mays, R, Jones, C., Holloway, G., and Studinski, D., iExperiences with Defect Prevention, i IBM Systems Journal, Vol. 29, No. 1, 1990.

6. Murphy, Brendan and Gent. Ted, iMeasuring System and Software Reliability Using and Automated Data Collection Processi, Quality and Reliability Engineering International CCC 0748-8017/95/050341-13pp., 1995.

7. Musa, John et. al., iSoftware Reliability; Measurement, Prediction, Application, McGraw-Hill, New Yrok, 1988.

8. Software Cosideration in Airbone Systems and Requirements Certification, Document Number RCTA/DO – 178B, Requirements and Technical Concepts for Aviation, RCTA Inc., Washington, DC, December 1, 1992.

9. System and Software Relaibility Assurance Notebook (draft), Prepared for Rome Laboratores by Peter Lakey, McDonnel Douglas Corporation and Ann Marie Neufelder, SoftRel, 1997.

4. Reliability Design Improvement Methods

4.1. Introduction

Reliability improvement can be attempted by many different methods. Among several methods of reliability improvement, design improvement is the most important. This is because the inherent system reliability is basically dependant on its design. Other methods such as maintenance, environmental control, etc. can only marginally improve systems reliability whereas; design improves substantially the systems reliability. Infact, most of the present days systems are designed for a predefined target reliability. This can be attempted either at the component level or at the system level. Wherever possible, component level design reliability improvement is preferred. However, there are technical, economic, and manufacturing limitations while attempting component level improvements. Therefore, design improvements at systems level are also necessary in many cases. This section discusses some of the design improvement techniques generally attempted by reliability engineers. Methods such as derating, redundancy, and stress reduction are discussed in details, and a method for reliability growth testing is discussed at the end of the section.

4.2. Derating

Reliability now a days has become part and parcel of the Electronic subsystems. One of the techniques of achieving enhanced reliability is Derating. Derating (Electrical Stress Analysis) is the reduction in electrical & thermal stresses applied to a part in order to decrease the part failure rate, which enhance the equipment reliability. Derating can be defined as follows: Operating the part at stresses value less than its rated value.

4.2.1. Importance of Derating

1. Derating is most effective tool for the designer to decrease the failure rates of part. Derating can help to compensate for many of the variables inherent in any design.
2. All electronic parts produced in an assembly line are not identical. Subtle differences & variations exist from one part to next. Proper part derating will

help to compensate for these part-to-part variations and minimize their impact up on the equipment reliability.

3. Electronic parts with identical manufacturer's part numbers may be purchased from a different suppliers. While these items are electrically interchanged there may be significant difference in design, material & manufacturing process. Derating will help to compensate for these differences.
4. The designer will try to anticipate the various electrical and environmental extremes to which the equipment may be subjected. If he fails to anticipate properly the impact of all of these variations, derating can provide an additional margin of safety.
5. It is also apparent that parts and their associated critical parameters are not completely stable over their entire life. Proper derating will help to assure that the circuit itself will continue to function properly in-spite of these part parameter changes.

4.2.2. Effect of Derating On Part Stress Reliability Prediction

During the useful life of an electronic part its reliability is a function of both the electrical & the thermal stresses to which the part is subjected. Increase in thermal stresses directly increases the junction temperature, which will increase failure rate according to the mathematical model of failure rate calculation. Also increasing the electrical stresses, results increase in failure rate. Both the stresses increase simultaneously failure rate & finally decreases the reliability.

Some parts are temperature sensitive so sometime failure occurs due to temperature. In such type of components, reduction in temperature by improvement in thermal design will result in reduced number of failures.

4.2.3. Method of Derating

All part's derating is done with reference to the absolute maximum ratings. The manufacturer in the specification or data sheet defines these ratings. Usually a part has several different absolute maximum ratings, such as voltage, current, power etc. Each of these absolute maximum ratings are unique, and must be applied individually and not in combination with any other absolute maximum rating. The absolute

maximum ratings state a maximum operating and/or storage temperature (junction or hotspot temperature) and various electrical values based upon DC power conditions measured in free air at 25 deg. C.

Derating must be cost effective. It should not be conservative to the point where the cost rises excessively. e.g. where lower than necessary part stresses are applied.

Derating can be accomplished either by reducing the stresses on the part or by increasing the strength of the part i.e. by selecting a part having greater strength. Actual derating procedures vary with different types of parts and their applications. Stress parameter for which the part should be derated will be different for different part categories. Details regarding the stress parameter along with the allowed stress factor for different types of parts are given in the subsequent sections.

Different type of component derated by different parameters such as Resistors are derated by power, by the ratio of the operating power to rated power & Capacitors are derated by reducing the applied voltage to the value lower than that for which the part is rated. Semiconductor is derated by limiting their power dissipation hence their junction temperature below the rated level.

4.3. Redundancy

Redundancy is another important method for system reliability improvement. This is a technique in which more number of components than actually required for operation is connected in parallel. There are many types of redundancies, viz. active, stand-by, k-out-of-n good system, etc. These techniques are discussed below:

4.3.1. Active Parallel Redundancy

This is the most commonly used redundancy by designers. This is also known as hot redundancy. Here, instead of using one component to do a function, we use 2, 3, or more number of components in parallel to do the same function. In this way, the components actively share the load among themselves effectively reducing the failure rate of each component. Therefore, the system failure rate reduces and reliability improves. As we increase the number of redundant components, we get diminishing returns after adding each additional component. That means, reliability improvement

by adding the first redundant component is more than that achieved by adding the second redundant component and so on.

4.3.2. Standby Redundancy

This is similar to the active parallel redundancy but each additional component used in parallel is connected through a switching (automatic or human operated) mechanism. That means, at any given system operational moment, only one component will be operating and other components are used only when the present active operational component fails. Therefore, each component takes full-connected load when it is in the operational mode, and takes zero loads when under standby mode. It should be kept in mind that reliability of switches used in standby redundancy affects the system reliability to a great extends. Therefore, reliability of switches must be very high.

4.3.3. K-out-of-M Redundancy

This is a type of redundancy in which a system is designed with M number of components. Out of these, it is essential that at least K of them be always in operational mode for achieving system success. (M-K) components are generally in standby mode. Whenever any active component fails, one of these standby components will become active and ensure system success.

It must be noted that the weight, cost, and volume of the system increases as a result of applying redundancy in engineering designs. Appropriate trade-offs are essential to optimize or maximize the effectiveness of applying redundancies in system design. Design engineers and managers must also keep in mind the following when attempting to adopt redundancy as a design tool for reliability improvement.

1) Redundancy is the easiest method of improving reliability
2) Any level of system reliability can be achieved
3) Volume, weight, cost increases in direct properties to the number of redundancies used
4) In some cases redundancy can be used only in higher levels of system

5) This should be used as a technique of reliability improvement when all other methods fail or are unacceptable due to technical or managerial reasons

6) Benefit/cost of redundancy is found to be best when we add the first redundancy. B/C ratio decreases for higher redundancies

4.4. Stress Reduction

Stress reduction is another design method for reliability improvement. Failure rate of component increases many times when the working environment or stress becomes more and more severe. This is basically because the material properties change with operating environment and as a result, the strength reduces. This leads to higher failure rates. For example for every 10^0C rise in temperature, failure rate of most electronic components becomes double. Humid and salty environment results in faster rates of corrosion and oxidation. Severe vibration, acceleration and shocks cause breakage, loose contacts, unbalance and change of control settings.

Let us now examine the reliability improvement concepts using the famous stress-strength concepts. As discussed in another section, both stress and strength follow some distribution. The interference area of stress-strength distributions represents the unreliability zone. That means reducing the interference area is a feasible method for reliability improvement. The interference area can be reduced by applying following techniques:

1. By increasing the gap between mean stress and mean strength
2. By reducing the variance of the stress-strength distributions
3. By a combination of 1 and 2

The first approach is basically similar to using a factor of safety or using the principle of derating. The second approach is either by controlling the variations in environment and applied load or by controlling the variations in strength by process control approach. This also illustrates that a better quality control mechanism will improve the product reliability. The third approach is by combining all the other approaches. Here appropriate trade-offs must be done for best results. Following figure illustrates all these concepts.

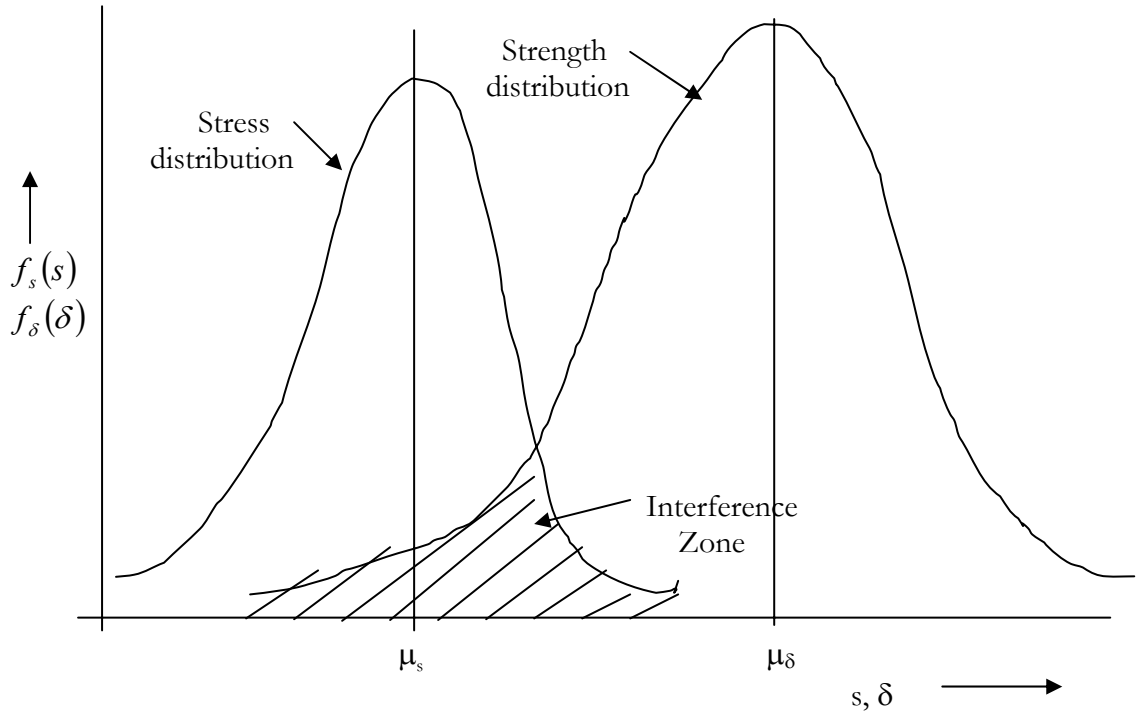


Figure 4.1: **Stress-strength interference**

4.4.1.1. Reliability Growth Testing

Prediction of reliability during the design and development stages is done using RGT. Here:

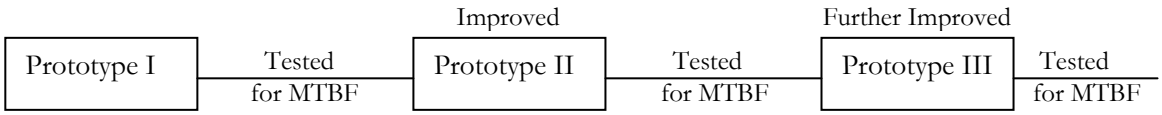


Figure 4.2: **Reliability Growth**

The design and manufacturing process is thus improved step by step. Value of MTBF indicates whether the design is improved or not at each stage. These prototype test data can be used for estimating MTBF or other reliability parameters for the final design. Duane originated this technique.

4.4.2. Duane Model

J.T. Duane developed this model in 1964. This model assumes that the failure times follow exponential density function (constant failure rate). He found that a plot of the cumulative number of failures per test time versus the logarithm of test time during growth testing approximately linear.

Let

T: total operation (test) time accumulated on all prototypes.
n(T): number of failures from the beginning of testing through time T.

After each failure occurs, systematic failure analysis is carried out and the system is modified after rectification. Life testing is again carried out on the improved product. If we plot $n(T)/T$ vs T on log-log paper, we get a straight line, for all types (electrical, mechanical, electronic, etc.) of equipments. From these empirical relationships, known as Duane plots, we can estimate MTBF of the system

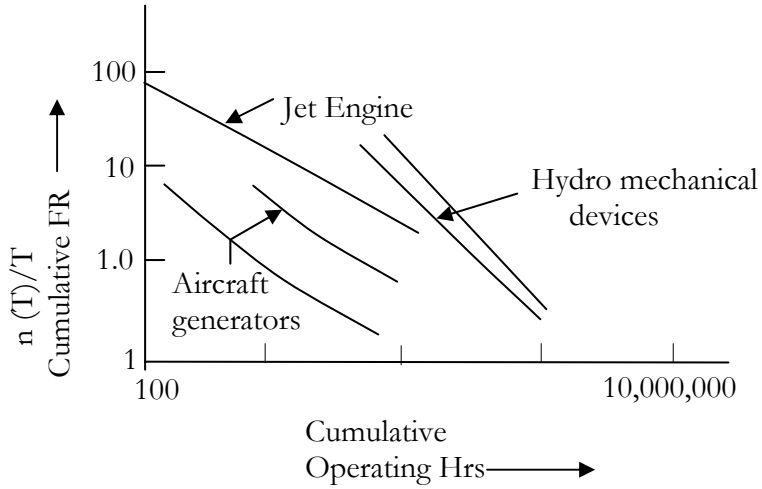


Figure 4.3: Duane Plots (on log-log paper) for Different Systems

Since Duane plots are straight lines,

$$\ln(n(T)/T) = b - \alpha \ln T$$

Solving for $n(T)$, we get

$$n(T) = KT^{1-\alpha},$$

Where,

$$K = e^b$$

Instantaneous FR, $\lambda(T) = \dfrac{dn(T)}{dT} = (1-\alpha)KT^{-\alpha}$

Instantaneous MTBF, $\mu(T) = \dfrac{1}{K(1-\alpha)}T^{\alpha}$

Since α is a positive number (≈ 0.5) this equation illustrates the growth of the MTBF, and reliability with accumulated test time.

4.5. Cumulative MTBF

$$\mu_c(T) = \frac{T}{n(T)} = \frac{T^{\alpha}}{K}$$

4.5.1. Alternate Duane Plot

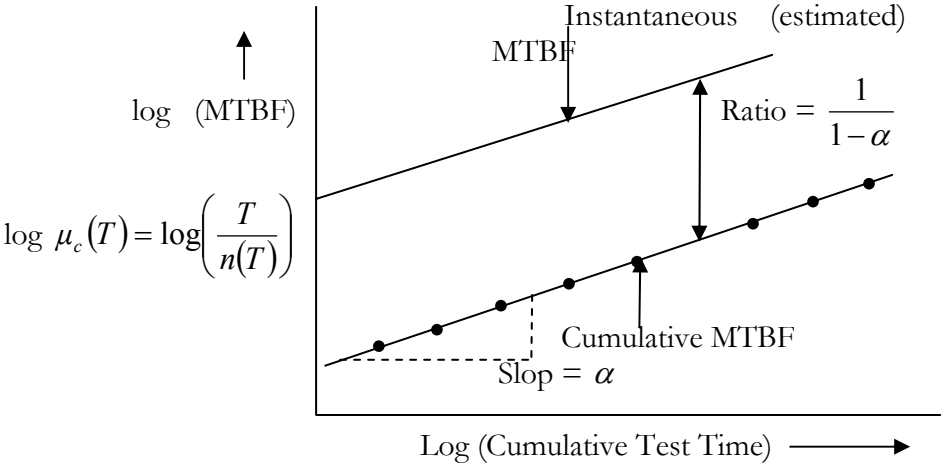


Figure 4.4: **Alternative Plot**

Plot $\log(T)$ vs $\log(MTBF)$. We get a straight-line $\alpha \rightarrow$ slope of this line.

Instantaneous MTBF, $\mu(T) = \frac{1}{1-\alpha}\mu_c(T)$ (the upper line).

Thus instantaneous MTBF can be directly obtained. Hence this plot is more useful.

4.5.2. Limitations

1. When failures may no longer be attributed to removable design defects, the growth of reliability may no longer be significant.
2. Refining the design to reduce random failures, or wear failures may become very expensive and thus unacceptable.
3. Reliability of mans produced items may be lower than that of tested prototypes.
4. Reliability of item in field service may be lower than those tested in laboratories.

To deal with these problems, we must concentrate on production quality control and realistically anticipate field condition.

EXAMPLE 5.1

A first prototype for a novel laser powered damage slicer is built. Failures occur at the following numbers of minutes: 1.1, 3.9, 8.2, 17.8, 79.7, 113.1, 208.4 and 239.1. After each failure the design is refined to avert further failures from the same mechanism. Determine the reliability growth coefficient α for the slicer.

Spreadsheet

Sl.No.	A N	B T	C ln(T)	D ln(n/T)
1	1	1.1	0.0953	-0.0953
2	2	3.9	1.3610	-0.6678
3	3	8.2	2.1041	-1.0055
4	4	17.8	2.8792	-1.4929
5	5	79.7	4.3783	-2.7688
6	6	113.1	4.7283	-2.9365
7	7	208.4	5.3395	-3.3935
8	8	239.1	5.4769	-3.3974

A least square fit made of column D versus column C. We obtain

$$\alpha = slope(D2:D9, C2:C9) = -0.654$$

$$\alpha = 0.654 \text{ (from equation } \ln(n(T)/T) = \cdots)$$

The straight line fit is quite good since the coefficient of determination is close to one; $r^2 = RSQ(D2:D9, C2:C9) = 0.988$.

5. Cost Analysis

5.1. Life Cycle Cost Analysis

One of the major considerations in establishing system reliability is life – cycle costs. Life-cycle costing is the process of determining all relevant costs from conceptual development through production, utilization, and phase-out. It is the total cost of ownership. Our interest in discussing life- cycle is to ensure that those costs affected by our choice of design variables, especially reliability (and later maintainability), are properly accounted for. There are many different ways to establish life-cycle costs categories; a typical cost element structure is shown in Table 5.1.

Table 5.1: Cost Categories

Acquisition cost	Operations and support costs	Phase -out
Research and development	Operations	Salvage value
Management	Facilities	Disposal costs
Engineering	Operators	
Design and prototyping	Consumables (energy and fuel)	
Engineering design	Unavailable time or downtime	
Fabrication	Support	
Testing and evaluation	Repair resources	
Production	Supply resources:	
Manufacturing	Repairables	
Plant facilities and overhead	Expendables	
Marketing and distribution	Tools, test, and support equipment	
	Failure costs	
	Training	
	Technical data	

In performing design trade-offs, total life-cycle costs of each alternative design should be estimated and compared. At the highest level, a life cycle cost model may take on the following form:

$$\text{Life-cycle cost} = \text{acquisition costs} + \text{operation costs} + \text{failure cost} + \text{support costs} - \text{net salvage value}$$

Where, Net salvage value = salvage value- disposal cost

Since the system will normally be operated over an extended period of time corresponding to its design life or economic life, the time value of money must be taken into account. The economic life is the number of years beyond which it is no longer economical to operate or maintain the system and replacement or discontinuance is justified on a cost basis. To discount monetary values over time all

revenues and costs can be expressed in present –day equivalent dollars. Therefore the following adjustments must be made. P is the present value and i is the real or effective, discount rate. If we assume a constant annual inflation rate of f and an annual return on investment rate of e, then $i \approx e - f$ for small values of f and e.

Let $P_F(i, d) = 1/(1+i)^d$ where F is a future amount at the end of year d, and $P_A(i, d) = [(1+i)^d - 1]/(1+i)^d]$, where A is an equal annual amount observed over d years. The term $P_A(i, d)$ is an annuity factor, which converts equal annual payments over d years to a single present –day equivalent amount. Writing Eq more explicitly,

$$\text{Life cycle cost} = C_u N + [F_o + P_A(i, t_d)\, C_o N] + \left[P_A(i,t_d) C_f \frac{t_0}{MTTF} N\right] + \left[F_S + P_A(i,t_d) C_s N\right] - \left[P_F(i,t_d) SN\right]$$

Where,

C_u = unit acquisition cost

N = number of identical units to be procured

F_o = fixed cost of operating

C_o = annual operating cost per unit

F_s = fixed support cost

C_s = annual support cost per unit

C_f = cost per failure

t_o = operating hours per year unit

t_d = design life (in years)

S = unit salvage value (a negative value is interpreted as a disposal cost)

The expression t_0/ MTTF in Eq is the expected number of failures per year assuming replacement or "repair to as good as new" condition of the failed unit (a renewal process which is discussed in the next chapter). The cost per failure C_f may be a repair cost, replacement cost, or a warranty cost. The unit acquisitions cost includes the design development and production costs allocated over the total number produced. As the reliability goal increases, these costs will increase because of additional reliability growth testing, improved manufacturing quality control, more expensive parts and material, increased use of redundancy, and additional resources committed to reliability improvement.

Assuming that only the unit acquisition cost and failure cost are sensitive to the design reliability, we may wish to compare the following expected present equivalent unit cost for each alternative:

$$C_u + (P/A, i, t_d)C_f - \frac{t_0}{MTTF}.$$

5.1.1. The Economics of Reliability and Maintainability and System Design

The reliability and maintainability program must pay for itself. How much reliability and maintainability should be designed into a product depends to a large degree on the cost (or profits) to be realized from the operational use of the product. In cost trade-off models were presented in order to relate R&M parameters to product life-cycle costs. The revenue and life-cycle cost model presented here is more comprehensive than the earlier models. Nevertheless, it is only an example of the many forms that such models may take. Our focus not surprisingly, is on those costs affected by the system reliability and maintainability.

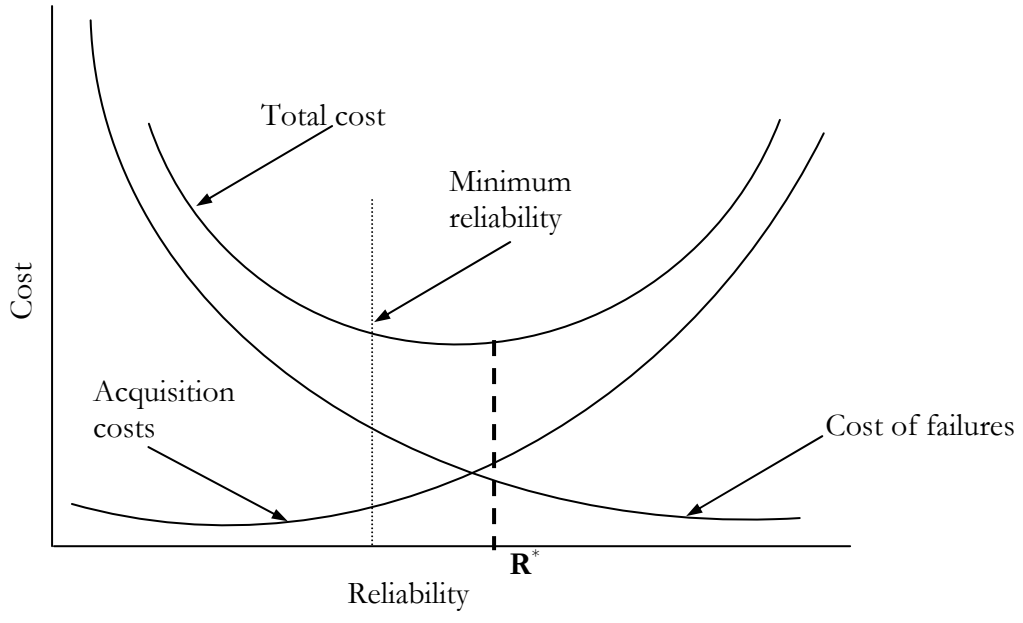


Figure 5.1: The total cost versus reliability curve

Figure 5.1 shows the total cost curve as the sum of the acquisition cost curve and the cost-of failures curve. Acquisition cost includes the cost of implementing and operating a reliability program in addition to the overall development and production

cost associated with the product. Acquisition cost consists of direct material and labor costs as well as indirect costs such as taxes, insurance, energy, production facilities and equipment, and overhead costs such as administrative, marketing, and product development costs. It is the product development that generally involves the engineering staff. Acquisition costs are increasing functions of reliability, not only because more organizational resources must be committed to achieve a higher reliability, but also because the material and production costs of the product must increase as well. This may be a result of more costly parts selection, added redundancy, stricter tolerances, excess strength, and increased quality control and inspection sampling during manufacture. The cost of failures may include warranty costs, liability costs, replacement or repair costs, the cost of the infrastructure necessary to support operational failures, and the loss of future profit (market share) as a result of loss of customer goodwill. These costs obviously decrease as reliability improves. The sum of the acquisition costs curve and the costs-of failures curve, shown in Figure 5.1, represents the desired reliability level. If the minimum –cost point exceeds critical reliability, for example, to meet a safety or contractual requirement, then it is desired level of reliability. Otherwise, the minimum reliability becomes the desired level. If a safety or liability cost associated with injury or loss of life can be quantified, it also can be included as a failure cost. Often, however, we are unable or unwilling to assign a cost to an injury or death, and we must be content to establish a lower bound on safety-related reliability parameters.

A similar cost curve exists as a function of the maintainability of a repairable product. However, it is more useful to consider the economics of a repairable system in terms of availability, a shown in Figure 5.2. With reliability fixed, as maintainability improves and restoration time decreases, system (operational) availability will increase. Therefore there will be less downtime and the costs, consisting of labor, facilities, equipment, and spares, and any loss of revenue associated with the operation of the system. Those acquisition and support costs that increase as the maintainability increases include the infrastructure necessary to implement the maintainability program; design and development costs associated with increased fault isolation, modularization, accessibility interchangeability and other design methods; higher salaries for increased maintenance skills levels; maintenance training; added repair capability; and increased availability of spare parts. To the extent that it

increases availability, the cost of a preventive maintenance program would be included in the category of increasing costs.

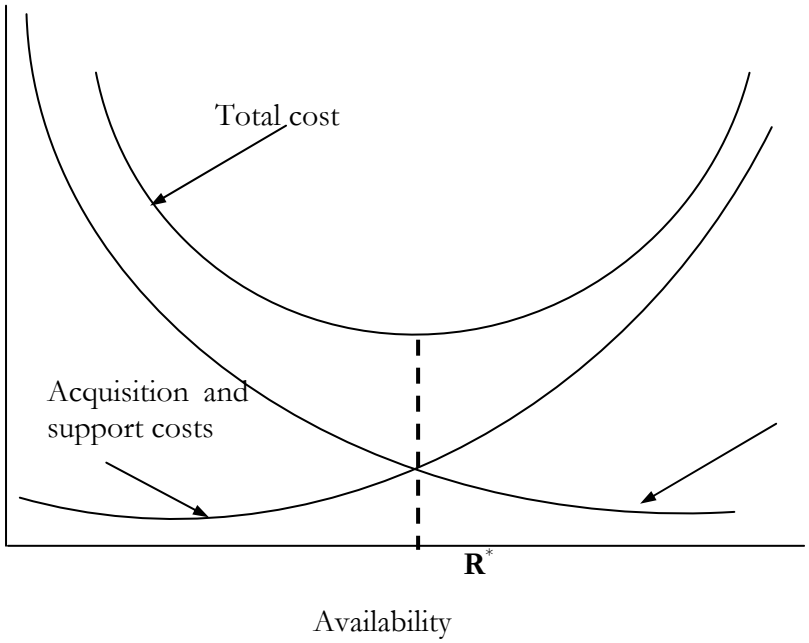


Figure 5.2: The total cost versus availability curve

5.1.2. Life-Cycle Cost Model

A generalization of the life-cycle cost given by Eq that takes the above cost elements into account is based on the following assumptions:

1. Failures resulting a renewal process (unit replacement or repair to as good as new condition)
2. All units are identical and are acquired at the same time (t = 0).
3. Annual operating requirements are constant.
4. The system is in a steady state (equilibrium).
5. There is no preventive maintenance.
6. No failures occur in standby, and perfect switching occurs with negligible down-time.

Mathematically, the life-cycle cost, LCC, can be expressed as

$$LCC(m,s,k,MTBF,MTTR,s_i,k_i) = C_u(MTBF,MTTR)(m+s) + F_0 + A_{sys}P_A(r,t_d)C_0 m + P_A(r,t_d)\frac{t_0}{MTBF}A_{sys}m(C_f + LMTTR) + F_{rep}k + P_A(r,t_d)C_{rep}k + \sum\left[C_i s_i + P_A(r,t_d)C_{rep}k_i\right] - P_F(r,t_d)S_a(m+s) \quad (5.1)$$

where,

C_u(MTB, MTTR) = unit acquisition cost

MTBF = the MTBF of the system failure distribution in operating hours

MTTR = repair or replacement time in hours

m = programmed number of operating units

s = number of spare units (standby redundancy)

k = number of repair channels

s_i = number of spares of component I

k_i = number of repair channels for component I

A_{sys}= effective system availability (average percentage of the m units operating)

F_0 = fixed cost of operating

C_0 = annual operating cost per unit

F_{rep} = initial acquisition cost per repair channel

$C_{rep,}$ = annual (support) cost per repair channel

C_f = fixed cost per failure

C_i = unit cost of component *I*

$C_{rep,i}$ = annual cost per repair channel for component *I*

L = labor rate ($ per hour)

t_0 = number of operating hours per year per unit.

t_d = design life (in years)

S_a = unit salvage value (a negative value is a disposal cost)

r = discount rate

$P_F(r, t_d)$ = $1/(1+r)^{td}$ is a present-value factor of a future amount at time t_d years at a discount rate of r.

$P_F(r, t_d)$ = $[(1+r)^{td} –1]/[r91+r)^{td}]$is a present-value factor of an annuity over t_d years at a discount rate of r.

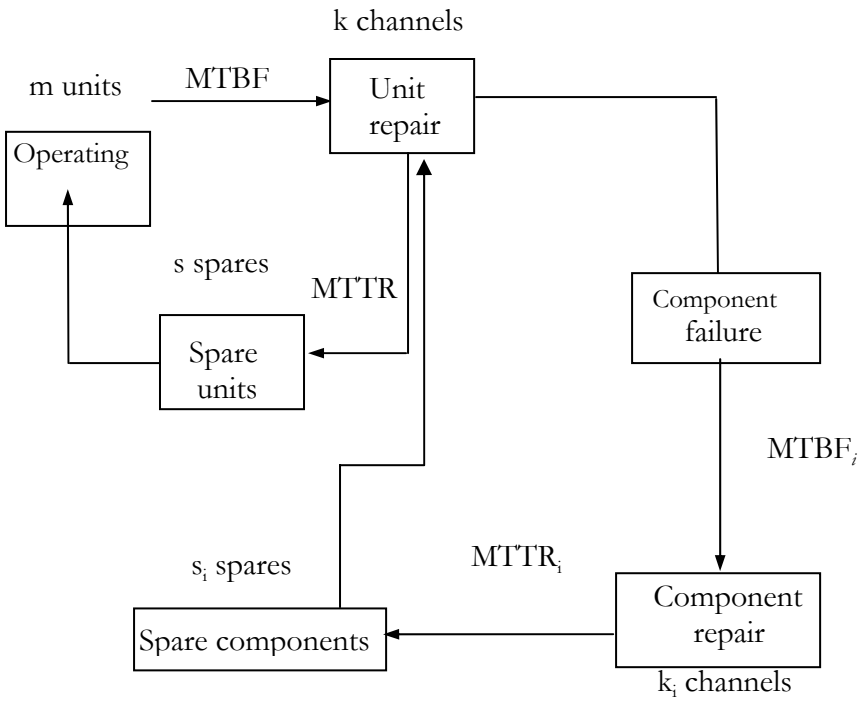


Figure 5.3: **Repair Cycle System**

The above cost model treats m, s, k, s_i, k_i, MTBF, and MTTR as design variables for repair –cycle system shown in Figure 5.3. The component mean time between failures ($MTBF_i$) and the component mean repair time. ($MTTR_i$) can be obtained from a reliability and maintainability allocation based on the computed values of the system MTBF and MTTR. Through the use of this model, trade-offs among these decision variables can be made. The unit acquisition cost, C_u, is assumed to be a function of the inherent reliability (MTBF) and maintainability (MTTR). This cost relationship is not written explicitly since it is very problem-specific and may take on many different functional forms. For example, if repair is accomplished by a fixed number of repair (labor) resources, k, the labor cost per failure, L, may be zero. On the other hand, if only replacement cost, C_f, is incurred when a failure occurs, there may be no repair channels necessary, or k = 0. For those components that are not repairable, k_i = 0, and an annual spares replenishment costs is incurred for those units that are discarded.

The effective system availability, A_{sys}, is a function of m, s, k, s_i, k_i, MTBF, and MTTR and is based on the concept of an operational availability That is, $A_{sys} = L_o/m$ where L_o is the expected number of units operating. If the failure times or repair times are not exponential, computer simulation can be used to find the steady-state

system availability as a function of number operating (m), number of standby spares (s), repair capability (k), number of component spares (s_i), and the inherent unit MTBF and MTTR.

In some applications it may be more desirable to compare alternative designs with regard to expected life-cycle profits rather than costs. Since profit = revenue – cost,

$$E[\text{profit}] = P_{A=}(r,t_d)A_{sys}Rm\text{-}LCC \tag{5.2}$$

Where R = revenue generated per operating unit per year. Although it is desirable to minimize Eq. (3.1) or maximize Eq (3.2), this can be difficult since m, s, k and s_i must be integers and the relationships are nonlinear. In most cases it is not possible to express A_{sys} in a simple closed form. On the other hand, given values for the design variables and the cost and revenue coefficients, the numerical evaluation of either equation should be straightforward.

5.2. Warranty Cost Analysis

Warranties are an important ingredient to competitive success. Effective warranty planning can ensure success, but lack of attention to cost analyses can spell disaster. This article is intended to introduce the basics of warranties and to identify sources for more information.

A warranty is the seller's assurance to the buyer that a product or service is as represented. An *express warranty* is one where the terms are explicitly stated in writing and an *implied warranty* is one where the seller automatically is responsible for the fitness of the product or service for use according to the Uniform Commercial Code.

Generally, three types of warranties are common for consumer goods: (1) the *ordinary free replacement* type, (2) the *unlimited free replacement* type, and (3) the *pro-rata* type. As the names imply, under the first two types, the seller provides a free replacement with the distinction between the two being that with type (1) the warranty on the replacement is for the remaining length of the original warranty while with type (2) it's for the same length as the original warranty. With the pro-rata type, the cost of the replacement depends on the age of the item at the time of replacement. Because the free replacement types seem to be most advantageous to the customer and the pro-rata most advantageous to the seller, a mixed policy type is

often used as a compromise. With this type, there's an initial period of free replacement, followed by a period of pro-rata policy.

Warranty planning includes a number of decisions starting with whether the product is repairable or non-repairable. The determining factor in deciding which applies to the product of interest usually depends on the ratio of the repair cost to the acquisition price. Once this decision is made, the type of warranty, the length of the warranty, and the funds required to cover the costs have to be determined. Often, the warranties offered by competitors in a particular product market weigh heavily on these decisions. At this point you may be wondering why warranties are a topic for the "Reliability Ques" series. It really shouldn't be a surprise because the single most important parameter in estimating the warranty cost is the rate at which the product is expected to fail. Usually, an initial estimate of the reliability based on predictions or similar products' experience is used to project costs, with a later switch to the analysis of actual warranty claims from an internal tracking or Failure Reporting and Corrective Action (FRACAS) type system.

Simple Warranty Example: Let's assume that a manufacturer of GPS devices plans to offer a 6-month warranty on the devices that cost Rs.100 each to produce. The expectation is to sell 10,000 devices and an internal test program indicates that the Mean-Time-To-Failure (MTTF) is 5 years after a stress-screening period. How much should the production cost be increased to cover the warranty cost?

W=6 months

C_0= RS. 100 (without warranty cost)

MTTF= 60 months

N= 10,000 units

The expected number of failures is:

$$F(t) = N\left[1 - e^{-(t/MTTF)}\right],$$

So that the number of failures over the interval dt is

$$df = \left(N/MTTF\right)e^{-t/MTTF}dt$$

The cost of failures is

$$C(t) = C_w\left[1 - \left(t/W\right)\right]$$

So that the warranty reserve cost is

$$WC = \int_0^W \left[\left(NC_W / MTTF \right) \left(1 - t/W \right) \right] e^{-t/MTTF} dt$$

Where C_W is the cost of including the warranty cost, so

$$WC = NC_W \left[1 - \left(MTTF / W \right) \left(1 - e^{-W/MTTF} \right) \right]$$

Therefore, the added warranty reserve fund per unit, per dollar cost C_W, is:

$$\left(WC/N \right) / C_W = 1 - \left(MTTF/W \right) \left(1 - e^{-W/MTTF} \right) = 1 - \left(60/6 \right) \left(1 - e^{-6/60} \right) = 0.0484$$

But, the production cost plus warranty cost,

$$C_W = C_0 + WC/N \quad \text{or}$$

$$C_0 = C_W \left[1 - \frac{WC/N}{C_W} \right] \quad \text{or}$$

$$C_W = \frac{C_0}{\left(1 - \frac{WC/N}{C_W} \right)} \quad \text{or}$$

$$C_W = \$100 / \left(1 - 0.0484 \right) = \$105.08$$

Therefore, the total warranty fund required would be:

$$WC = \left(\frac{WC/N}{C_W} \right) x N x C_W = (0.0484)(10{,}000)(\$105.08) = \$50{,}833$$

6. Accelerated Life Testing Data Analysis

6.1. Introduction

Accelerated life testing consists of tests designed to quantify the life characteristics of a product, component or system under normal use conditions by testing the units at higher stress levels in order to accelerate the occurrence of failures. Performed correctly, these tests can provide valuable information about a product's performance under use conditions that can empower a manufacturer to bring its products to market more quickly and economically than would be possible using standard life testing methods.

Accelerated life tests are component life tests with components operated at high stresses and failure data observed. While high stress testing can be performed for the sole purpose of seeing where and how failures occur and using that information to improve component designs or make better component selections, we will focus in this section on accelerated life testing for the following two purposes:

1. To study how failure is accelerated by stress and fit an acceleration model to data from multiple stress levels
2. To obtain enough failure data at high stress to accurately project (extrapolate) what the CDF at use will be.

Test planning and operation for a (multiple) stress cell life test experiment consists of the following:

- Pick several combinations of the relevant stresses (the stresses that accelerate *the failure mechanism under investigation*). Each combination is a "stress cell". Note that you are planning for only one mechanism of failure at a time. Failures on test due to any other mechanism will be considered censored run times.
- Make sure stress levels used are not too high - to the point where new failure mechanisms that would never occur at use stress are introduced. Picking a maximum allowable stress level requires experience and/or good engineering judgment.

- Put random samples of components in each stress cell and run the components in each cell for fixed (but possibly different) lengths of time.
- Gather the failure data from each cell and use the data to fit an acceleration model and a life distribution model and use these models to project reliability at use stress conditions.

In typical life data analysis, the practitioner analyzes life data of a product's sample operating under normal conditions in order to quantify the life characteristics of the product and make predictions about all of the products in the population. For a variety of reasons, manufacturers may wish to obtain reliability results for their products more quickly than they can with data obtained under normal operating conditions. Instead, they may use quantitative accelerated life tests to capture life data for the product under accelerated stress conditions, which cause the products to fail more quickly. Quantitative accelerated life tests (QALT) are designed to quantify the life of the product and produce the data required for accelerated life data analysis. This analysis method uses life data obtained under accelerated conditions to extrapolate an estimated probability density function (*pdf*) for the product under normal use conditions.

QALT tests can employ usage rate acceleration or overstress acceleration to speed up the time-to-failure for the products under test. With usage rate acceleration, which is appropriate for products that do not operate continuously under normal conditions, the analyst operates the products under test at a greater rate than normal to simulate longer periods of operation under normal conditions. Data from this type of test can be analyzed with standard life data analysis techniques. With overstress acceleration, one or more environmental factors that cause the product to fail under normal conditions (like temperature, voltage, humidity, etc.) are increased in order to stimulate the product to fail more quickly. Data from this type of test require special accelerated life data analysis techniques, which include a mathematical model to "translate" the overstress probability density functions to normal use conditions. The analysis techniques for data from quantitative overstress accelerated life tests are discussed.

6.2. Data and Data Types

The analysis of accelerated tests relies extensively on data. Specifically, analysis relies on life and stress data or times-to-failure data at a specific stress level. The accuracy of any prediction is directly proportional to the quality of and accuracy of the supplied data. Good data, along with the appropriate distribution and life-stress model, usually results in good predictions. Bad or insufficient data will always result in bad predictions.

For the purposes of this reference, we will separate data into two types based on the failure or success of the product. Failure data will be referred to as complete data and success data will be referred to as suspended (or right censored) data. In other words, we know that a product failed after a certain time (complete data) or we know that it operated successfully up to a certain time (suspended or right censored data). Each type is explained next.

6.2.1. Complete Data

Most non-life data, as well as some life data, are what we refer to as complete data. Complete data means that the value of each sample unit is observed (or known). For example, if we had to compute the average test score for a sample of 10 students, complete data would consist of the known score for each student. For products, known times-to-failure (along with the stress level), comprise what is usually referred to as complete data. For example, if we tested five units and they all failed, we would then have complete information as to the time-to-failure for each unit in the sample.

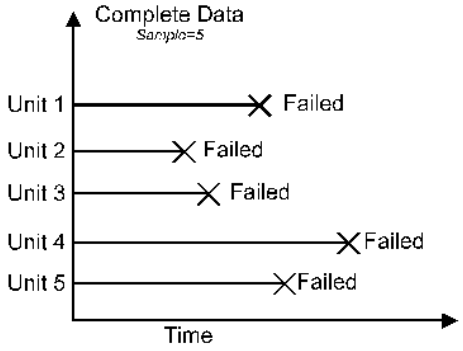


Figure 6.1: **Complete Data**

6.2.2. Censored Data

It is also possible that some of the units have not yet failed when the life data are analyzed. This type of data is commonly called right censored data, or suspended

data. Assume that we tested five units and three failed. In this scenario, our data set is composed of the times-to-failure of the three units that failed (complete data) and the running time of the other two units that have not failed at the time the data are analyzed (suspended data). This is the most common censoring scheme and it is used extensively in the analysis of field data.

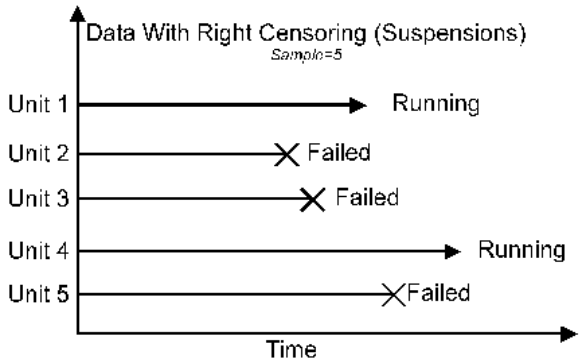


Figure 6.2: Censored Data

6.2.2.1. Censored Type I Data

During the T hours of test we observe *r* failures (where *r* can be any number from 0 to *n*). The (exact) failure times are t_1, t_2, ..., t_r and there are (*n* - *r*) units that survived the entire T-hour test without failing. Note that T is fixed in advance and *r* is random, since we don't know how many failures will occur until the test is run. Note also that we assume the exact times of failure are recorded when there are failures.

This type of censoring is also called "right censored" data since the times of failure to the right (i.e., larger than T) are missing. Another (much less common) way to test is to decide in advance that you want to see exactly *r* failure times and then test until they occur. For example, you might put 100 units on test and decide you want to see at least half of them fail. Then *r* = 50, but T is unknown until the 50th fail occurs. This is called Censored Type II data.

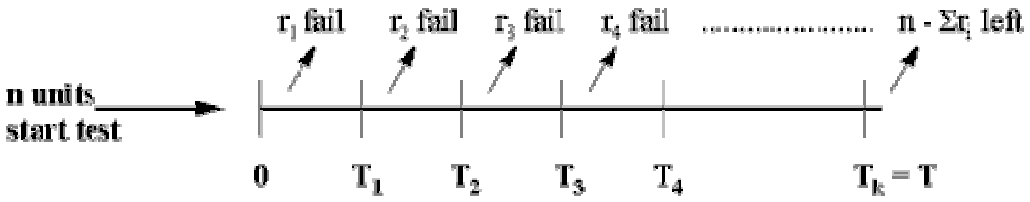
6.2.2.2. Censored Type II Data

We observe t_1, t_2, ..., t_r, where *r* is specified in advance. The test ends at time T = t_r, and (*n*-*r*) units have survived. Again we assume it is possible to observe the exact time of failure for failed units.

Type II censoring has the significant advantage that you know in advance how many failure times your test will yield - this helps enormously when planning

adequate tests. However, an open-ended random test time is generally impractical from a management point of view and this type of testing is rarely seen.

Sometimes exact times of failure are not known; only an interval of time in which the failure occurred is recorded. This kind of data is called **Readout** or **Interval** data and the situation is shown in the figure below:



6.2.2.3. Multi-censored Data

In the most general case, every unit observed yields exactly one of the following three types of information:

- a run-time if the unit did not fail while under observation
- an exact failure time
- an interval of time during which the unit failed.

6.3. Stress Types and Stress Levels

In an effective quantitative accelerated life test, the analyst chooses one or more stress types that cause the product to fail under normal use conditions. Stress types can include temperature, voltage, humidity, vibration or any other stress that directly affects the life of the product. He/she then applies the stress(es) at various increased levels and measures the times-to-failure for the products under accelerated test conditions. For example, if a product normally operates at 290K and high temperatures cause the product to fail more quickly, then the accelerated life test for the product may involve testing the product at 310K, 320K and 330K in order to stimulate the units under test to fail more quickly. In this example, the stress type is temperature and the accelerated stress levels are 310K, 320K and 330K. The use stress level is 290K. Using the life data obtained at each accelerated stress level, the analyst can use standard life data analysis techniques to estimate the parameters for the life distribution (*e.g.* Weibull, exponential or lognormal) that best fits the data at each stress level. This results in an overstress probability density function (*pdf*) for

each accelerated stress level. Another mathematical model, the life-stress relationship, is then required to estimate the probability density function (*pdf*) at the normal use stress level based on the characteristics of the *pdfs* at each accelerated stress level.

The application of the stress (under test conditions and/or during normal use) can be constant (time-independent) or time-dependent. When the stress is constant, the stress level applied to a sample of units does not vary with time. Each unit is tested under the same accelerated temperature for the duration of the test. For example, ten units are tested at 310K for 100 hours, ten different units are tested at 320K for 100 hours and ten different units are tested at 330K for 100 hours.

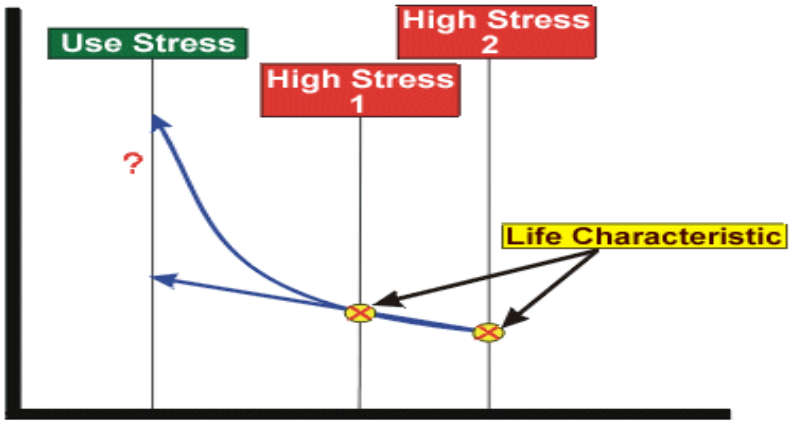
When the stress is time-dependent, the stress applied to a sample of units varies with time. Time-dependent stresses can be applied in a variety of ways. For example, if temperature is the stress type, each unit may be tested at 310K for 10 hours then increased to 320K for 10 hours then increased to 330K for 10 hours over the duration of the test. Alternatively, the units may be placed in a test chamber where the temperature starts at 310K and increases by five degrees every ten minutes until the chamber reaches 330K. Some common types of time-dependent stress profiles include step-stress, ramp-stress and various profiles in which the application of the stress is a continuous function of time. Figure 1 and Figure 2 display two examples of the many time-dependent stress profiles that can be used in an accelerated life test design.

6.4. Life-Stress relationships

Statisticians, mathematicians and engineers have developed life-stress relationship models that allow the analyst to extrapolate a use level probability density function (*pdf*) from life data obtained at increased stress levels. These models describe the path of a life characteristic of the distribution from one stress level to another. The life characteristic can be any life measure, such as the mean or median, expressed as a function of stress. For example, for the Weibull distribution, the scale parameter, η (eta), is considered to be stress-dependent and the life-stress model for data that fits the Weibull distribution is assigned to eta.

You must choose a life-stress relationship that fits the type of data being analyzed. Available life-stress relationships include the Arrhenius, Eyring, and inverse

power law models. These models are designed to analyze data with one stress type (*e.g.* temperature, humidity, or voltage). The temperature-humidity and temperature-nonthermal relationships are combination models that allow you to analyze data with two stress types (*e.g.* temperature and voltage or temperature and humidity). The general log-linear and proportional hazards models can be used to analyze data where up to eight stress types need to be considered. Finally, the cumulative damage (or cumulative exposure) model has been developed to analyze data where the application of the stress (either at the accelerated stress levels or at the use stress level) varies with time



6.5. Analyzing Data from Accelerated Life Tests

Using the life data obtained at each accelerated stress level, standard life data analysis techniques can be used to estimate the parameters for the life distribution (e.g. Weibull, exponential or lognormal) that best fits the data at each stress level. This results in an overstress probability density function (*pdf*) for each accelerated stress level. Another mathematical model, the life-stress relationship, is then required to estimate the probability density function (*pdf*) at the normal use stress level based on the characteristics of the *pdf*s at each accelerated stress level. The plot in Figure 6.3 demonstrates the relationship between life and stress for a particular product.

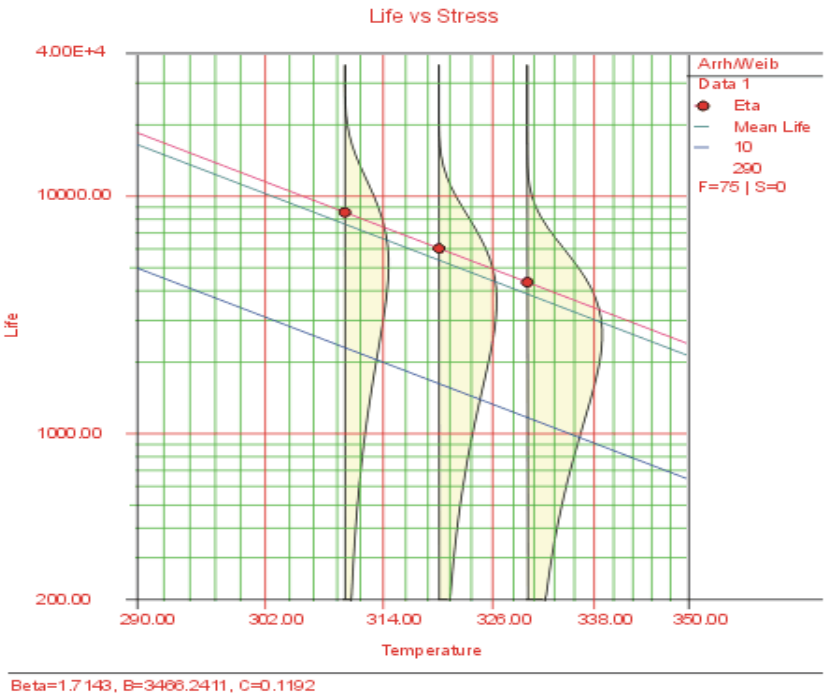


Figure 6.3: **The relationship between life and stress.**

6.6. How do you fit an acceleration model?

Once a life distribution and a life-stress relationship have been selected, the parameters (*i.e.* the variables that govern the characteristics of the *pdf*) need to be determined. Several parameter estimation methods, including probability plotting, least squares and maximum likelihood, are available.

As with estimating life distribution model parameters, there are two general approaches for estimating acceleration model parameters:

- Graphical estimation (or computer procedures based on a graphical approach)
- Maximum Likelihood Estimation (an analytic approach based on writing the likelihood of all the data across all the cells, incorporating the acceleration model)

Another promising method of fitting acceleration models is sometimes possible when studying failure mechanisms characterized by a stress-induced gradual degradation process that causes the eventual failure. This approach fits models based on degradation data and has the advantage of not actually needing failures. This

overcomes censoring limitations by providing measurement data at consecutive time intervals for every unit in every stress cell.

6.6.1. Graphical Method

Graphical analysis is the simplest method for obtaining results in both life data and accelerated life testing analyses. Although they have limitations in general graphical methods are easily implemented and easy to interpret.

The graphical method for estimating the parameters of accelerated life data involves generating two types of plots. First, the life data at each individual stress level are plotted on a probability paper appropriate to the assumed life distribution (*i.e.* Weibull, exponential, or lognormal). The parameters of the distribution at each stress level are then estimated from the plot. Once these parameters have been estimated at each stress level, the second plot is created on a paper that linearizes the assumed life-stress relationship (*i.e.* Arrhenius, inverse power law, etc.). The parameters of the life-stress relationship are then estimated from the second plot. The life distribution and life-stress relationship are then combined to provide a single model that describes the accelerated life data

6.6.1.1. Life Distribution Parameters at Each Stress Level

The first step in the graphical analysis of accelerated data is to calculate the parameters of the assumed life distribution at each stress level. Because life data are collected at each test stress level in accelerated life tests, the assumed life distribution is fitted to data at each individual stress level. The parameters of the distribution at each stress level are then estimated using the probability plotting method described next.

6.6.1.2. Life Distribution Probability Plotting

The easiest parameter estimation method (to use by hand) for complex distributions, such as the Weibull distribution, is the method of probability plotting. Probability plotting involves a physical plot of the data on specially constructed probability plotting paper. This method is easily implemented by hand as long as one can obtain the appropriate probability plotting paper.

Probability plotting looks at the *cdf* (cumulative density function) of the distribution and attempts to linearize it by employing a specially constructed paper. For example, in the case of the 2-parameter Weibull distribution, the *cdf* and unreliability $Q(T)$ can be shown to be,

$$F(T) = Q(T) = 1 - e^{-\left(\frac{T}{\eta}\right)^{\beta}}$$

This function can then be linearized (*i.e.* put into the common form of $y = a + bx$) as follows,

$$\begin{aligned} Q(T) &= 1 - e^{-\left(\frac{T}{\eta}\right)^{\beta}} \\ \ln(1 - Q(T)) &= \ln\left(e^{-\left(\frac{T}{\eta}\right)^{\beta}}\right) \\ \ln(1 - Q(T)) &= -\left(\frac{T}{\eta}\right)^{\beta} \\ \ln(-\ln(1-Q(T))) &= \beta \ln\left(\frac{T}{\eta}\right) \\ \ln\left(\ln\left(\frac{1}{1-Q(T)}\right)\right) &= \beta \ln(T) - \beta \ln(\eta) \end{aligned}$$

Then setting,

$$y = \ln\left(\ln\left(\frac{1}{1 - Q(T)}\right)\right)$$

and, the equation can be rewritten as,

$$y = \beta x - \beta \ln(\eta)$$

Which, is now a linear equation with a slope of β and an intercept of $-\beta \ln(\eta)$.

The next task is to construct a paper with the appropriate *x*- and *y- axes*. The *x-axis* is easy since it is simply logarithmic. The *y-axis*, however, must represent,

$$y = \ln\left(\ln\left(\frac{1}{1 - Q(T)}\right)\right)$$

Where, $Q(T)$ is the unreliability. Such papers have been created by different vendors and are called Weibull probability plotting papers.

To illustrate, consider the following probability plot on a Weibull Probability Paper

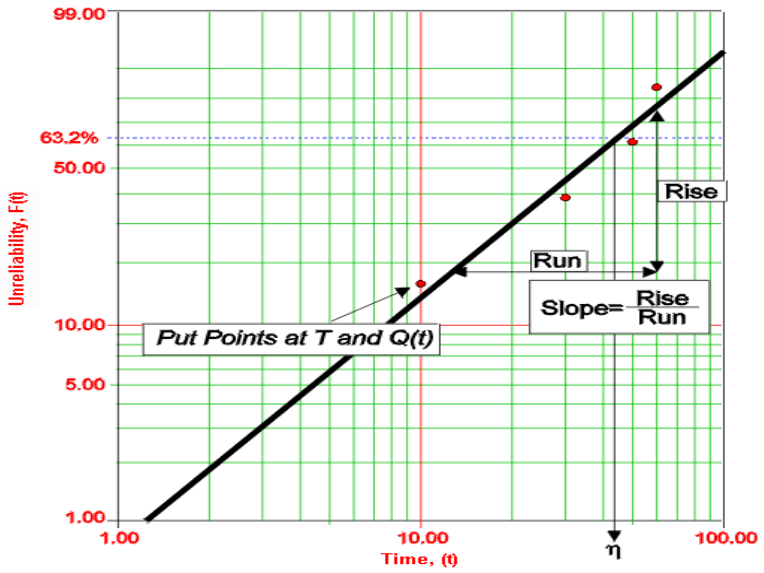


Figure 6.4: Weibull Probability Paper

This paper is constructed based on the *y* and *x* transformation mentioned previously where the *y-axis* represents unreliability and the *x-axis* represents time. Both of these values must be known for each point (or time-to-failure) we want to plot.

Then, given the *y* and *x* value for each point, the points can easily be placed on the plot. Once the points are placed on the plot, the best possible straight line is drawn through these points. Once the line is drawn, the slope of the line can be obtained (most probability papers include a slope indicator to facilitate this) and thus the parameter β, which is the value of the slope, can be obtained.

To determine the scale parameter, η(also called the characteristic life by some authors), a little more work is required. Note that from before,

$$Q(T) = 1 - e^{-\left(\frac{T}{\eta}\right)^{\beta}}$$

so at $T = \eta$

$$\begin{aligned} Q(T) &= 1 - e^{-\left(\frac{\eta}{\eta}\right)^{\beta}} \\ &= 1 - e^{-1} \\ &= 0.632 \\ &= 63.2\% \end{aligned}$$

Thus if we entered the *y axis* at $Q(T)$ = 63.2%, the corresponding value of T will be equal to η. Using this simple, but rather time-consuming methodology, then,

the parameters of the Weibull distribution can be determined. For data obtained from accelerated tests, this procedure is repeated for each stress level.

6.6.1.3. Determining the X and Y Position of the Plot Points

The points plotted on the probability plot represent our data, or more specifically in life data analysis, times-to-failure data. So if we tested four units that failed at 10, 20, 30 and 40 hours at a given stress level, we would use these times as our *x* values or time values. Determining the appropriate *y* plotting position, or the unreliability, is a little more complex. To determine the *y* plotting positions, we must first determine a value called the median rank for each failure.

6.6.1.4. Median Ranks

Median ranks are used to obtain an estimate of the unreliability, $Q(T_j)$ for each failure. It represents the value that the true probability of failure, $Q(T_j)$, should have at the j^{th} failure out of a sample of *N* units, at a 50% confidence level. This is an estimate of the value based on the binomial distribution. The rank can be found for any percentage point, *P*, greater than zero and less than one, by solving the cumulative binomial distribution for *Z* (rank for the j^{th} failure).

$$P=\sum_{k=j}^{N}\binom{N}{k}Z^k(1-Z)^{N-k} \tag{6.1}$$

Where, *N* is the sample size and *j* the order number.

The median rank is obtained by solving the following equation for

$$0.50=\sum_{k=j}^{N}\binom{N}{k}Z^k(1-Z)^{N-k}$$

For example if $N = 4$ and we have four failures at that particular stress level, we would solve the median rank equation, Eqn. (A), four times; once for each failure with $j = 1, 2, 3$ and 4, for the value of *Z*. This result can then be used as the unreliability for each failure, or the *y* plotting position. Solution of equation (6.1) requires numerical methods.

A more straightforward and easier method of estimating median ranks is to apply two transformations to Eqn. (6.1), first to the beta distribution and then to the F distribution, resulting in ,

$$MR = \frac{1}{1+\frac{N-j+1}{j}F_{0.50;m;n}}$$
$$m = 2(N-j+1)$$
$$n = 2j$$

$F_{0.50;m;n}$ denotes the F distribution at the 0.50 point, with m and n degrees of freedom, for the j^{th} failure out of N units. A quick and less accurate approximation of the median ranks is also given by,

$$MR = \frac{j-0.3}{N+0.4}$$

6.6.1.5. Some Shortfalls of Manual Probability Plotting

Besides the most obvious shortfall of probability plotting, the amount of effort required, manual probability plotting is not always consistent in the results. Two people plotting a straight line through a set of points will not always draw this line the same way and they will therefore come up with slightly different results. In addition, when dealing with accelerated test data a probability plot must be constructed for each stress level. This implies that sufficient failures must be observed at each stress level, which is not always possible.

6.6.1.6. Life-Stress Relationship Plotting

Once the parameters of the life distribution have been obtained using probability plotting methods, a second plot is created in which life is plotted versus stress. To do this, a life characteristic must be chosen to be plotted. The life characteristic can be any percentile, such as $B(x)$ life, the scale parameter, mean life, etc. The plotting paper used is a special type of paper that linearizes the life-stress relationship. For example, a log-log paper linearizes the inverse power law relationship, and a log-reciprocal paper linearizes the Arrhenius relationship. The parameters of the model are then estimated by solving for the slope and the intercept of the line. This methodology is illustrated in Example 1.

EXAMPLE 1

Consider the following times-to-failure data at three different stress levels.

Stress	393 psi	408 psi	423 psi
Time Failed (hrs)	3450	3300	2645
	4340	3720	3100
	4760	4180	3400
	5320	4560	3800
	5740	4920	4100
	6160	5280	4400
	6580	5640	4700
	7140	6233	5100
	8101	6840	5700
	8960	7380	6400

Estimate the parameters for a Weibull assumed life distribution and for the inverse power law life-stress relationship.

SOLUTION

First the parameters of the Weibull distribution need to be determined. The data is individually analyzed (for each stress level) using the probability plotting method, or software such as ReliaSoft's Weibull++, with the following results:

$$[\hat{\beta}_1 = 3.8,\ \hat{\eta}_1 = 6692]$$
$$[\hat{\beta}_2 = 4.2,\ \hat{\eta}_2 = 5716]$$
$$[\hat{\beta}_3 = 4,\ \hat{\eta}_3 = 4774]$$

Where,

$\hat{\beta}_1, \hat{\eta}_1$ are the parameters of the 393 psi data.
$\hat{\beta}_2, \hat{\eta}_2$ are the parameters of the 408 psi data.
$\hat{\beta}_3, \hat{\eta}_3$ are the parameters of the 423 psi data.

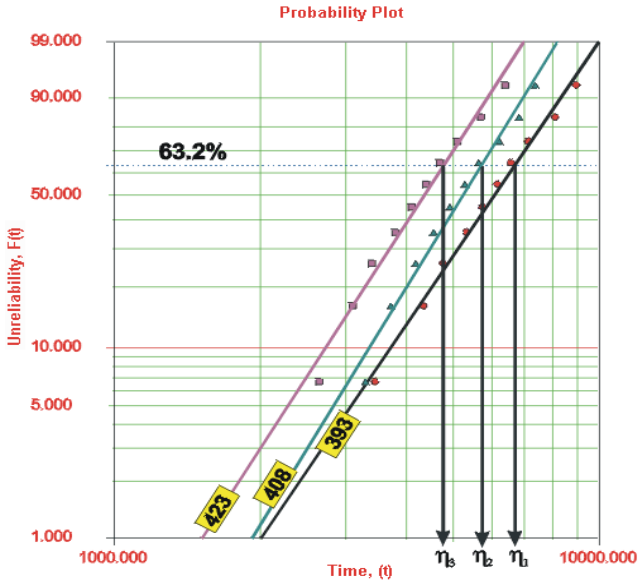


Figure 6.5: **Probability Plot**

Since the shape parameter, $\hat{\beta}$ is not common for the three stress levels, the average value is estimated.

$$\hat{\beta}_{common} = 4$$

Averaging the *betas* is one of many simple approaches available. One can also use a weighted average, since the uncertainty on beta is greater for smaller sample sizes. In most practical applications the value of $\hat{\beta}$ will vary (even though it is assumed constant) due to sampling error, etc. The variability in the value of $\hat{\beta}$ is a source of error when performing analysis by averaging the *betas*. MLE analysis, which uses a common $\hat{\beta}$, is not susceptible to this error. MLE analysis is the method of parameter estimation used in ALTA and it is explained in the next section.

Redraw each line with a $\hat{\beta}$ = 4 and estimate the new *eta's* as follows.

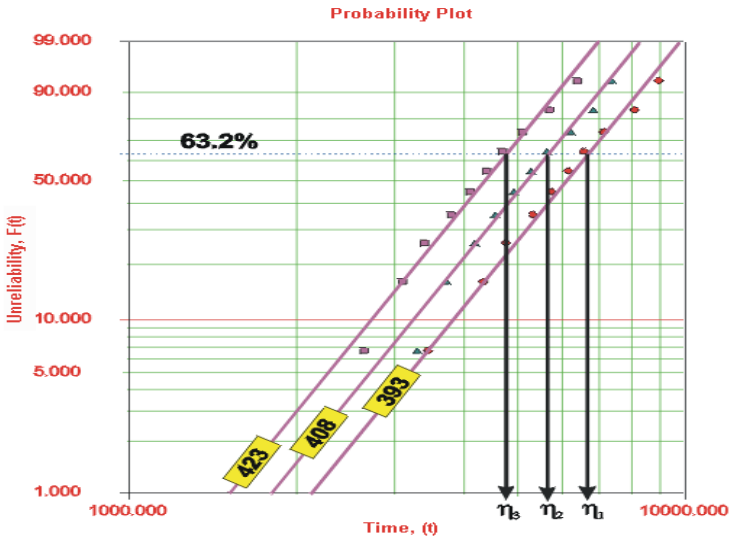


Figure 6.6: Probability Plot

$\hat{\eta}_1$ = 6650

$\hat{\eta}_2$ = 5745

$\hat{\eta}_3$ = 4774.

The IPL relationship is given by:

$$L(V) = \frac{1}{KV^n}$$

Where, L represents a quantifiable life measure (η in the Weibull case), V represents the stress level, K is one of the parameters and n is another model parameter. The relationship is linearized by taking the logarithm of both sides, which yields,

$$\ln(L) = -\ln K - n \ln V$$

Where, L = η, (-ln$K$) is the intercept, and (-n) is the slope of the line. The values of η obtained previously are now plotted on a log-linear scale yielding the following plot,

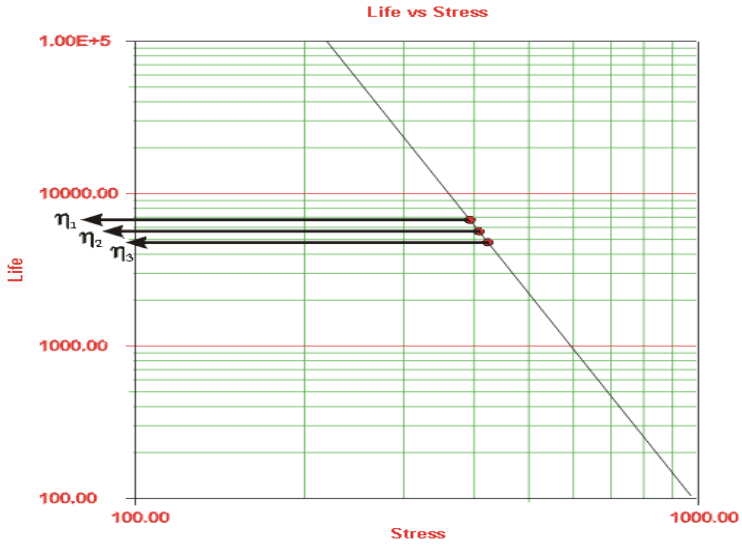


Figure 6.7: **Life versus Stress**

The slope of the line is the ηparameter, which is obtained from the plot:

$$\begin{aligned} Slope &= \frac{\ln(T_2) - \ln(T_1)}{\ln(V_2) - \ln(V_1)} \\ &= \frac{\ln(10,000) - \ln(6,000)}{\ln(360) - \ln(403)} \\ &= -4.5272 \end{aligned}$$

Thus,

$$\hat{n} = 4.5272$$

Solving the inverse power law equation with respect to K yields,

$$\hat{K} = \frac{1}{LV^n}$$

Substituting V = 403, the corresponding L (from the plot), L = 6,00 and the previously estimated n,

$$\begin{aligned} \hat{K} &= \frac{1}{6000\ 403^{4.5272}} \\ &= 2.67 \cdot 10^{-16} \end{aligned}$$

6.6.1.7. How to fit an Arrhenius Model with Graphical Estimation

Graphical methods work best (and are easiest to describe) for a simple one-stress model like the widely used Arrhenius model:

$$t_f = A\exp\left\{\frac{\Delta H}{kT}\right\}$$

with T denoting temperature measured in degrees Kelvin (273.16 + degrees Celsius) and k is Boltzmann's constant (8.617×10^{-5} in eV/°K).

When applying an acceleration model to a distribution of failure times, we interpret the deterministic model equation to apply at any distribution percentile we want. This is equivalent to setting the life distribution scale parameter equal to the model equation (T_{50} for the lognormal, α for the Weibull and the MTBF or $1/\lambda$ for the exponential). For the lognormal, for example, we have

$$T_{50} = Ae^{\frac{\Delta H}{kT}}$$

$$\ln T_{50} = y = \ln A + \Delta H\left(\frac{1}{kT}\right)$$

This can be written as

$$y = a + bx \quad \text{with } b = \Delta H \text{ and } x = \frac{1}{kT}$$

So, if we run several stress cells and compute T_{50}'s for each cell, a plot of the natural log of these T_{50}'s versus the corresponding $1/k$T values should be roughly linear with a slope of ΔH and an intercept of ln A. In practice, a computer fit of a line through these points is typically used to obtain the Arrhenius model estimates. There are even commercial Arrhenius graph papers that have a temperature scale in $1/k$T units and a T_{50} scale in log units, but it is easy enough to make the transformations and then use linear or log-linear papers.

That T is in Kelvin in the above equations. For temperature in Celsius, use the following for $1/k$T: 11605/(TCELSIUS + 273.16)

An example will illustrate the procedure.

Graphical Estimation: An Arrhenius Model Example

Component life tests were run at 3 temperatures: 85°C, 105°C and 125°C. The lowest temperature cell was populated with 100 components; the 105° cell had 50

components and the highest stress cell had 25 components. All tests were run until either all the units in the cell had failed or 1000 hours was reached. Acceleration was assumed to follow an Arrhenius model and the life distribution model for the failure mode was believed to be lognormal. The normal operating temperature for the components is 25°C and it is desired to project the use CDF at 100,000 hours.

TEST RESULTS:

Cell 1 (85°C): 5 failures at 401, 428, 695, 725 and 738 hours. 95 units were censored at 1000 hours running time.

Cell 2 (105°C): 35 failures at 171, 187, 189, 266, 275, 285, 301, 302, 305, 316, 317, 324, 349, 350, 386, 405, 480, 493, 530, 534, 536, 567, 589, 598, 599, 614, 620, 650, 668, 685, 718, 795, 854, 917, and 926 hours. 15 units were censored at 1000 hours running time.

Cell 3 (125°C): 24 failures at 24, 42, 92, 93, 141, 142, 143, 159, 181, 188, 194, 199, 207, 213, 243, 256, 259, 290, 294, 305, 392, 454, 502 and 696. 1 unit was censored at 1000 hours running time.

Failure analysis confirmed that all failures were due to the same failure mechanism (if any failures due to another mechanism had occurred, they would have been considered censored run times in the Arrhenius analysis).

Steps to Fitting the Distribution Model and the Arrhenius Model

1. Do graphical plots for each cell and estimate T_{50}'s and sigma's.
2. Put all the plots on the same sheet of graph paper and check whether the lines are roughly parallel (a necessary consequence of true acceleration).
3. If satisfied from the plots that both the lognormal model and the constant sigma from cell to cell are consistent with the data, plot the cell ln T_{50}'s versus the 11605/(TCELSIUS + 273.16) cell values, check for linearity and fit a straight line through the points. Since the points have different degrees of precision, because different numbers of failures went into their calculation, it is recommended that the number of failures in each cell be used as weights in a regression program, when fitting a line through the points.

4. Use the slope of the line as the $\hat{H}$ estimate and calculate the Arrhenius A constant from the intercept using $A = e^{intercept}$.

5. Estimate the common sigma across all the cells by the weighted average of the individual cell sigma estimates. Use the number of failures in a cell divided by the total number of failures in all cells as that cells weight. This will allow cells with more failures to play a bigger role in the estimation process.

6.6.1.8. Comments on the Graphical Method

Although the graphical method is simple, it is quite laborious. Furthermore, many issues surrounding its use require careful consideration. Some of these issues are presented next:

- What happens when no failures are observed at one or more stress level? In this case, plotting methods cannot be employed. Discarding the data would be a mistake since every piece of life data information is important. (In other words, no failures at one stress level combined with observed failures at other stress level(s) are an indication of the dependency of life on stress. This information cannot be discarded.)

- In the step at which the life-stress relationship is linearized and plotted to obtain its parameters, you must be able to linearize the function, which is not always possible.

- In real accelerated tests the data sets are small. Separating them and individually plotting them and then subsequently replotting the results, increases the underlying error.

- During initial parameter estimation, the parameter that is assumed constant will more than likely vary. What value do you use?

- Confidence intervals on all of the results cannot be ascertained using graphical methods.

6.6.2. MLE (Maximum Likelihood) Parameter Estimation

The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability (likelihood) of the sample data. From a

statistical point of view, the method of maximum likelihood is considered to be more robust (with some exceptions) and yields estimators with good statistical properties. In other words, MLE methods are versatile and apply to most models and to different types of data. In addition, they provide efficient methods for quantifying uncertainty through confidence bounds. Although the methodology for maximum likelihood estimation is simple, the implementation is mathematically intense. Using today's computer power, however, mathematical complexity is not a big obstacle. The MLE methodology is presented next.

6.6.2.1. Background Theory

This section presents the theory that underlies maximum likelihood estimation for complete data. If x is a continuous random variable with *pdf,*

$$f(x;\theta_1,\theta_2,...,\theta_k)$$

Where, $\theta_1, \theta_2, ..., \theta_k$ are k unknown constant parameters which need to be estimated, conduct an experiment and obtain N independent observations, $x_1, x_2, ..., x_N$. Then the likelihood function is given by the following product,

$$L(x_1,x_2,...,x_N|\theta_1,\theta_2,...,\theta_k)=L=\prod_{i=1}^{N} f(x_i;\theta_1,\theta_2,...,\theta_k)$$
$$i=1,2,...,N$$

The logarithmic likelihood function is given by:

$$\Lambda=\ln L=\sum_{i=1}^{N}\ln f(x_i;\theta_1,\theta_2,...,\theta_k)$$

The maximum likelihood estimators (MLE) of $\theta_1, \theta_2, ..., \theta_k$ are obtained by maximizing L or Λ.

By maximizing Λ, which is much easier to work with than L, the maximum likelihood estimators (MLE) of $\theta_1, \theta_2, ..., \theta_k$ are the simultaneous solutions of k equations such that,

$$\frac{\partial(\Lambda)}{\partial\theta_j}=0,\quad j=1,2,...,k$$

Even though it is common practice to plot the MLE solutions using median ranks (points are plotted according to median ranks and the line according to the

MLE solutions), this is not completely accurate. As it can be seen from the equations above, the MLE method is independent of any kind of ranks or plotting methods. For this reason, many times the MLE solution appears not to track the data on the probability plot. This is perfectly acceptable since the two methods are independent of each other and in no way suggests that the solution is wrong.

6.6.2.2. Illustrating the MLE Method Using the Exponential Distribution

- To estimate $\hat{\lambda}$, for a sample of *n* units (all tested to failure), first obtain the likelihood function,

$$\begin{aligned} L(\lambda|t_1, t_2, ..., t_n) &= \prod_{i=1}^{n} f(t_i) \\ &= \prod_{i=1}^{n} \lambda e^{-\lambda t_i} \\ &= \lambda^n \cdot e^{-\lambda \sum_{i=1}^{N} t_i} \end{aligned}$$

- Take the natural log of both sides,

$$\Lambda = \ln(L) = n \ \ln(\lambda) - \lambda \sum_{i=1}^{n} t_i$$

- Obtain $\frac{\partial \Lambda}{\partial \lambda}$, and set it equal to zero,

$$\frac{\partial \Lambda}{\partial \lambda} = \frac{n}{\lambda} - \sum_{i=1}^{n} t_i = 0$$

- Solve for $\hat{\lambda}$or,

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} t_i}$$

Notes on lambda

- Note that the value of $\lambda$is an estimate because if we obtain another sample from the same population and re-estimate λ, the new value would differ from the one previously calculated.
- In plain language, $\hat{\lambda}$is an estimate of the true value of λ.
- How close is the value of our estimate to the true value? To answer this question, one must first determine the distribution of the parameter, in this

case λ. This methodology introduces a new term, confidence level, which allows us to specify a range for our estimate with a certain confidence level.

- The treatment of confidence intervals is integral to reliability engineering and to all of statistics.

6.6.2.3. Illustrating the MLE Method Using the Normal Distribution

To obtain the MLE estimates for the mean, $\bar{T}$ and standard deviation, σ_T for the normal distribution, start with the *pdf* of the normal distribution which is given by:

$$f(T) - \frac{1}{\sigma_T\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{T-\bar{T}}{\sigma_T}\right)^2}$$

If $T_1, T_2, \ldots T_N$ are known times-to-failure (and with no suspensions), then the likelihood function is given by:

$$L(T_1, T_2, \ldots, T_N | \bar{T}, \sigma_T) = L = \prod_{i=1}^{N}\left[\frac{1}{\sigma_T\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{T_i-\bar{T}}{\sigma_T}\right)^2}\right]$$

$$L = \frac{1}{(\sigma_T\sqrt{2\pi})^N} e^{-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{T_i-\bar{T}}{\sigma_T}\right)^2}$$

Then,

$$\Lambda = \ln L = -\frac{N}{2}\ln(2\pi) - N\ln\sigma_T - \frac{1}{2}\sum_{i=1}^{N}\left(\frac{T_i-\bar{T}}{\sigma_T}\right)^2$$

Then taking the partial derivatives of Λ with respect to each one of the parameters and setting it equal to zero yields,

$$\frac{\partial(\Lambda)}{\partial\bar{T}} = \frac{1}{\sigma_T^2}\sum_{i=1}^{N}(T_i - \bar{T}) = 0 \quad \text{(B)}$$

and,

$$\frac{\partial(\Lambda)}{\partial\sigma_T} = -\frac{N}{\sigma_T} + \frac{1}{\sigma_T^3}\sum_{i=1}^{N}(T_i - \bar{T})^2 = 0 \quad \text{(C)}$$

Solving Eqns. (B) and (C) simultaneously yields,

$$\bar{T} = \frac{1}{N}\sum_{i=1}^{N} T_i$$

And,

$$\hat{\sigma}_T^2 = \frac{1}{N}\sum_{i=1}^{N}(T_i - \bar{T})^2$$

$$\hat{\sigma}_T = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(T_i - \bar{T})^2}$$

These solutions are only valid for data with no suspensions, *i.e.* all units are tested to failure. In cases in which suspensions are present, the methodology changes and the problem becomes much more complicated.

6.6.2.4. Estimator

As mentioned above, the parameters obtained from maximizing the likelihood function are estimators of the true value. It is clear that the sample size determines the accuracy of an estimator. If the sample size equals the whole population, then the estimator is the true value. Estimators have properties such as unbiasedness, sufficiency, consistency and efficiency. Numerous books and papers deal with these properties and this coverage is beyond the scope of this reference. However, we would like to briefly address unbiasedness and consistency.

6.6.2.5. Unbiased Estimator

An estimator is said to be unbiased if and only if the estimator $\hat{\theta} = d(X_1, X_2, \ldots, X_N)$ satisfies the condition $E[\hat{\theta}] = \theta$ for all $\theta \in \Omega$. Note that $E[X]$ denotes the expected value of X and is defined by (for continuous distributions),

$$E[X] = \int_{\varpi} x \cdot f(x)dx$$
$$X \in \varpi$$

This implies that the true value is not consistently underestimated nor overestimated by an unbiased estimator.

6.6.3. Conclusions

Two methods for estimating the parameters of accelerated life testing models were presented. First, the graphical method was illustrated using a probability plotting method for obtaining the parameters of the life distribution. The parameters of the life-stress relationship were then estimated graphically by linearizing the model. However, not all life-stress relationships can be linearized. In addition, estimating the

parameters of each individual distribution leads to an accumulation of uncertainties, depending on the number of failures and suspensions observed at each stress level. Furthermore, the slopes (shape parameters) of each individual distribution are rarely equal (common). Using the graphical method, one must estimate a common shape parameter (usually the average) and repeat the analysis. By doing so, further uncertainties are introduced on the estimates and these are uncertainties that cannot be quantified. On the other hand, treating both the life distribution and the life-stress relationship as one model, the parameters of that model can be estimated using the complete likelihood function. Doing so, a common shape parameter is estimated for the model, thus eliminating the uncertainties of averaging the individual shape parameters. All uncertainties are accounted for in the form of confidence bounds, which are quantifiable because they are obtained based on the overall model.

6.7. Calculated Results and Plots

Once you have calculated the parameters to fit a life distribution and a life-stress relationship to a particular data set, you can obtain the same plots and calculated results that are available from standard life data analysis. Some additional results, related to the effects of stress on product life, are also available. In addition, for the failure rate, reliability/unreliability and *pdf* plots, the information can be displayed for a given stress level in a two-dimensional plot or for a range of stress levels in a three-dimensional plot (*e.g.* failure rate vs. time vs. stress). Some frequently used metrics include:

- **Reliability Given Time:** The probability that a product will operate successfully at a particular point in time under normal use conditions. For example, there is an 88% chance that the product will operate successfully after 3 years of operation at a given stress level.
- **Probability of Failure Given Time:** The probability that a product will be failed at a particular point in time under normal use conditions. Probability of failure is also known as "unreliability" and it is the reciprocal of the reliability. For example, there is a 12% chance that the product will be failed after 3 years of operation at a given stress level (and an 88% chance that it will operate successfully).

- **Mean Life:** The average time that the products in the population are expected to operate at a given stress level before failure. This metric is often referred to as mean time to failure (MTTF) or mean time before failure (MTBF).
- **Failure Rate:** The number of failures per unit time that can be expected to occur for the product at a given stress level.
- **Probability Plot:** A plot of the probability of failure over time. This can display either the probability at the use stress level or, for comparison purposes, the probability at each test stress level. *(Note that probability plots are based on the linearization of a specific distribution. Consequently, the form of a probability plot for one distribution will be different than the form for another. For example, an exponential distribution probability plot has different axes than that of a normal distribution probability plot.)*
- **Reliability vs. Time Plot:** A plot of the reliability over time at a given stress level. A similar plot, unreliability vs. time, is also available.
- **pdf Plot:** A plot of the probability density function (*pdf*) at a given stress level.
- **Failure Rate vs. Time Plot:** A plot of the failure rate over time at a given stress level. This can display the instantaneous failure rate at a given stress level in a two-dimensional plot or the failure rate vs. time vs. stress in a three-dimensional plot.
- **Life vs. Stress Plot:** A plot of the product life vs. stress. A variety of life characteristics, like B(10) life or eta, can be displayed on the plot. This plot demonstrates the effect of a particular stress on the life of the product.
- **Standard Deviation vs. Stress Plot:** A plot of the standard deviation vs. stress, which provides information about the spread of the data at each stress level.
- **Acceleration Factor vs. Stress Plot:** A plot of the acceleration factor vs. stress.

6.7.1. Examples of Reporting for Parametric Data Analysis

Following are some examples of the information that can be generated using parametric data analysis. While this is by no means complete, it serves as a starting point for the information that can be obtained with the proper collection of data and parametric analysis.

6.7.1.1. Probability Plot

Probability plotting was originally a method of graphically estimating distribution parameter values. With the use of computers that can precisely calculate parametric values, the probability plot now serves as a graphical method of assessing the goodness of fit of the data to a chosen distribution. Probability plots have nonlinear scales that will essentially linearize the distribution function, and allow for assessment of whether the data set is a good fit for that particular distribution based on how close the data points come to following the straight line. The y-axis usually shows the unreliability or probability of failure, while the x-axis shows the time or ages of the units. Specific characteristics of the probability plot will change based on the type of distribution.

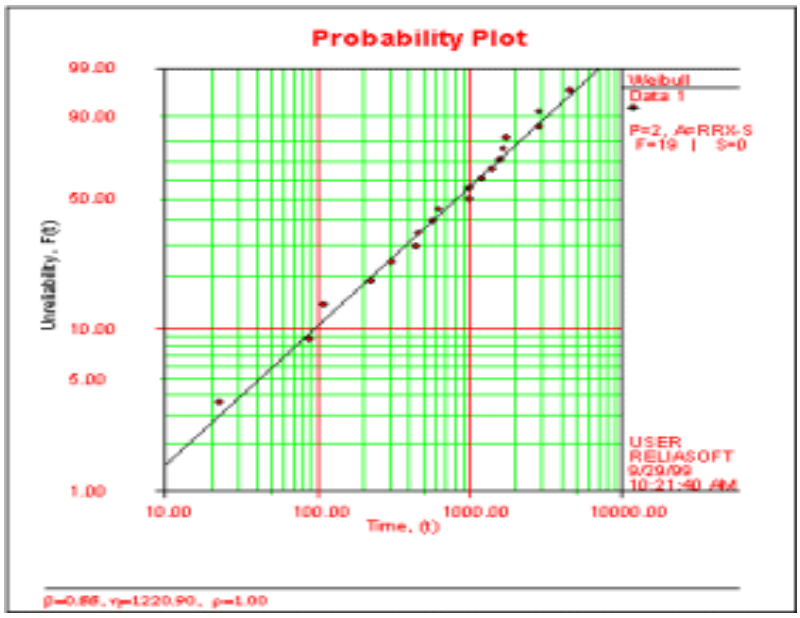


Figure 6.8: Probability Plot

6.7.1.2. Reliability Function

The reliability function gives the continuous probability of a successful mission versus the time of the mission. This is similar to the probability plot in that it shows

the performance of the product versus the time. However, it does not have nonlinear scales on the axes and the y-axis gives the reliability instead of the unreliability.

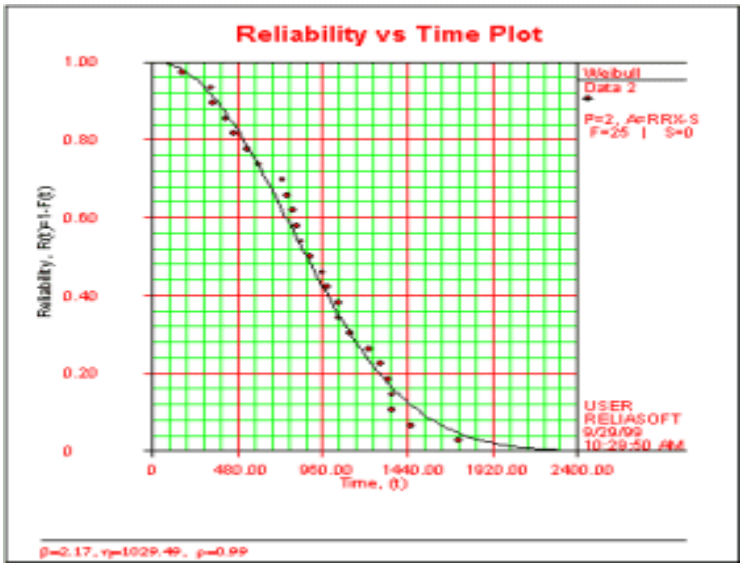


Figure 6.9: Reliability versus Time

6.7.1.3. Probability Density Function

The probability density function (*pdf*) represents the relative frequency of failures with respect to time. It basically gives a description of how the entire population from which the data is drawn is spread out over time or usage. The probability density function is most commonly associated with the "bell curve," which is the shape of the *pdf* of the normal or Gaussian distribution.

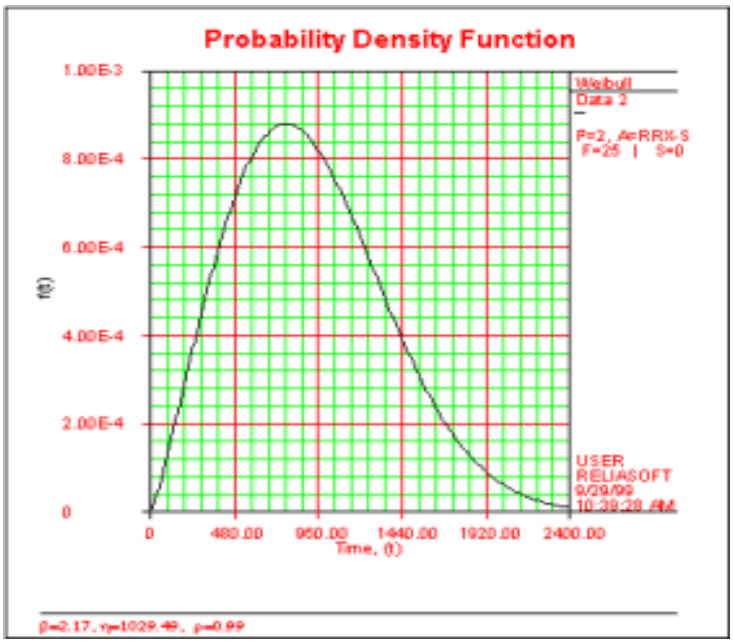


Figure 6.10: Probability Density Function

6.7.1.4. Failure Rate Function

The failure rate function indicates how the number of failures per unit time of the product changes with time. This provides a measure of the instantaneous probability of product failure changes as usage time is accumulated. The failure rate plot is associated with the "bathtub curve," which is an amalgamation of different failure rate curves to illustrate the different ways in which products exhibit failure characteristics over the course of their lifetimes.

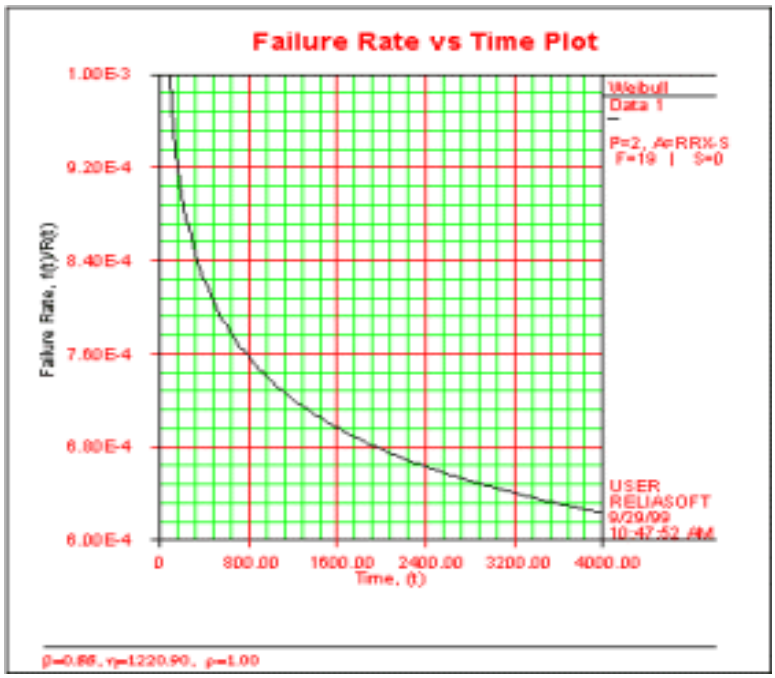


Figure 6.11: Failure Rate vs Time

6.7.1.5. Life vs. Stress Plot

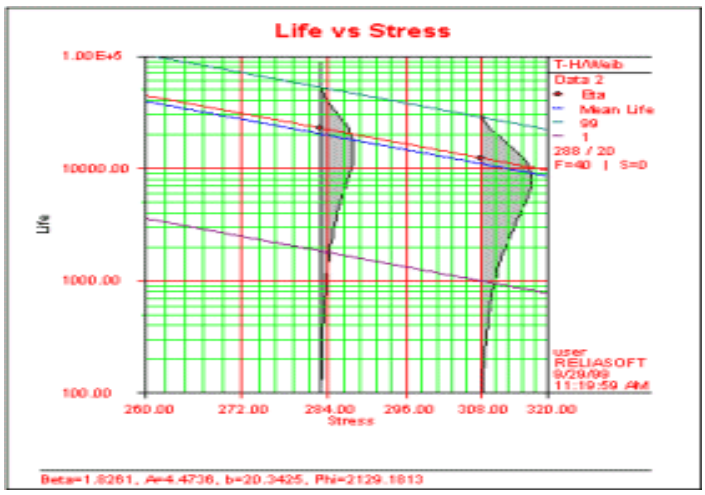


Figure 6.12: Life vs. Stress Plot

The Life vs. Stress plot is a product of accelerated life testing or reliability testing that is performed at different stress levels. This indicates how the life performance of the product changes at different stress levels. The gray shaded areas are actually *pdf* plots for the product at different stress levels. Note that it is difficult to make a complete graphical comparison of the *pdf* plots due to the logarithmic scale of the y-axis.

6.7.1.6. Reliability Growth

Reliability growth is an important component of a reliability engineering program. It essentially models the change in a product's reliability over time and allows for projections on the change in reliability in the future based on past performance. It is useful in tracking performance during development and aids in the allocation of resources. There are a number of different reliability growth models available that are suitable to a variety of data types. The above chart is a graphical representation of the logistic reliability growth model.

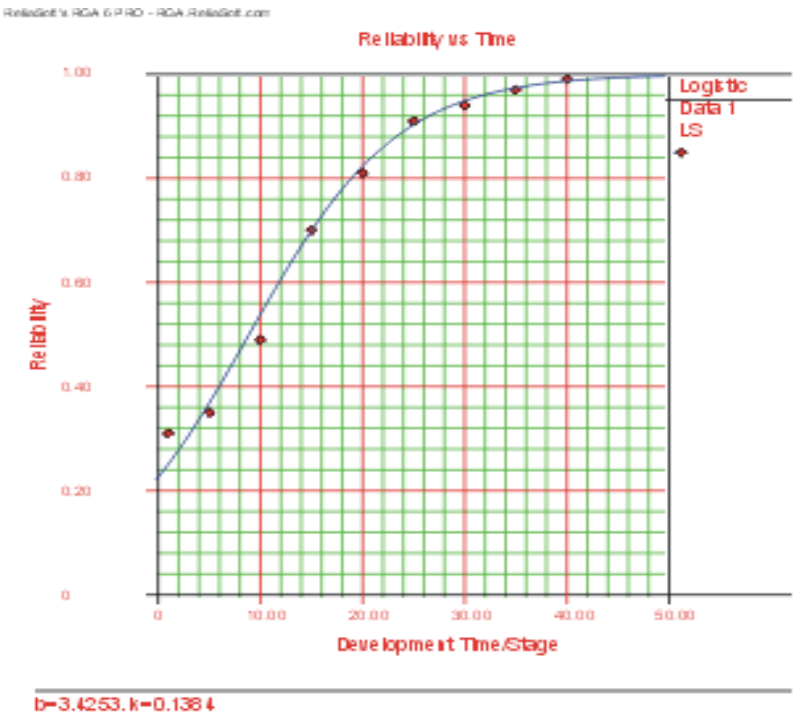


Figure 6.13: Reliability Growth

6.8. Confidence Bounds

Because life data analysis results are estimates based on the observed lifetimes of a product's sample, there is uncertainty in the results due to the limited sample sizes. Confidence bounds (also called confidence intervals) are used to quantify this uncertainty due to sampling error by expressing the confidence that a specific interval

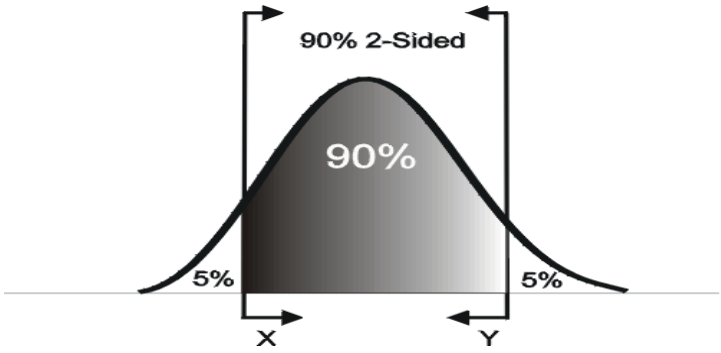
contains the quantity of interest. Whether or not a specific interval contains the quantity of interest is unknown.

Confidence bounds can be expressed as two-sided or one-sided. Two-sided bounds are used to indicate that the quantity of interest is contained within the bounds with a specific confidence. One-sided bounds are used to indicate that the quantity of interest is above the lower bound or below the upper bound with a specific confidence. Depending on the application, one-sided or two-sided bounds are used. For example, the analyst would use a one-sided lower bound on reliability, a one-sided upper bound for percent failing under warranty and two-sided bounds on the parameters of the distribution. (*Note that one-sided and two-sided bounds are related. For example, the 90% lower two-sided bound is the 95% lower one-sided bound and the 90% upper two-sided bounds is the 95% upper one-sided bound.*)

6.8.1. One-Sided and Two-Sided Confidence Bounds

Confidence bounds (or intervals) are generally described as one-sided or two-sided.

6.8.1.1. Two-Sided Bounds

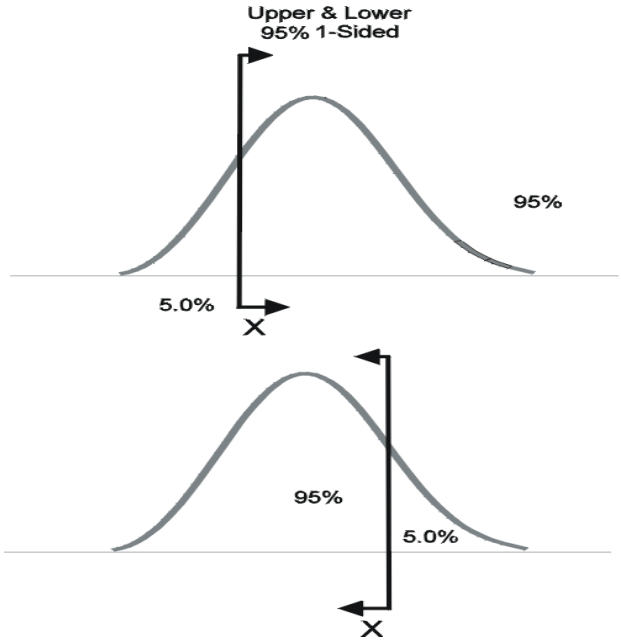


When we use two-sided confidence bounds (or intervals) we are looking at where most of the population is likely to lie. For example, when using 90% two-sided confidence bounds, we are saying that 90% lies between *X* and *Y*, with 5% less than *X* and 5% greater than *Y*.

6.8.1.2. One-Sided Bounds

When using one-sided intervals, we are looking at the percentage of units that are greater or less (upper and lower) than a certain point *X*.

For example, 95% one-sided confidence bounds would indicate that 95% of the population is greater than *X* (if 95% is a lower confidence bound) or that 95% is less than *X* (if 95% is an upper confidence bound).



In ALTA, (from Reliasoft) we use upper to mean the higher limit and lower to mean the lower limit, regardless of their position, but based on the value of the results. So for example, when returning the confidence bounds on the reliability, we would term the lower value of reliability as the lower limit and the higher value of reliability as the higher limit. When returning the confidence bounds on probability of failure, we will again term the lower numeric value for the probability of failure as the lower limit and the higher value as the higher limit.

6.8.1.3. Electronic Devices Example

Twelve electronic devices were put into a continuous accelerated test. The accelerated stresses were temperature and voltage, with use level conditions of 328K and 2V respectively. The data obtained is shown in the table below:

Time-to-Failure (hr)	Temperature K	Voltage V
620	348	3
632	348	3
658	348	3
822	348	3
380	348	5
416	348	5
460	348	5
596	348	5
216	378	3
246	378	3
332	378	3
400	378	3

Do the following:

1. Using the T-NT Weibull model analyze the data in ALTA and determine the MTTF and *B*(10) life for these devices at use level. Determine the upper and lower 90% 2-sided confidence intervals on the results.

2. Examine the effects of each stress on life.

3. Figures 6.14 and 6.15 below examine the effects of each stress on life, while Figure 6.15 examines the effects of the combined stresses on the reliability. Specifically, Figure 6.14 below shows the life vs. voltage plot with temperature held constant at 328K.

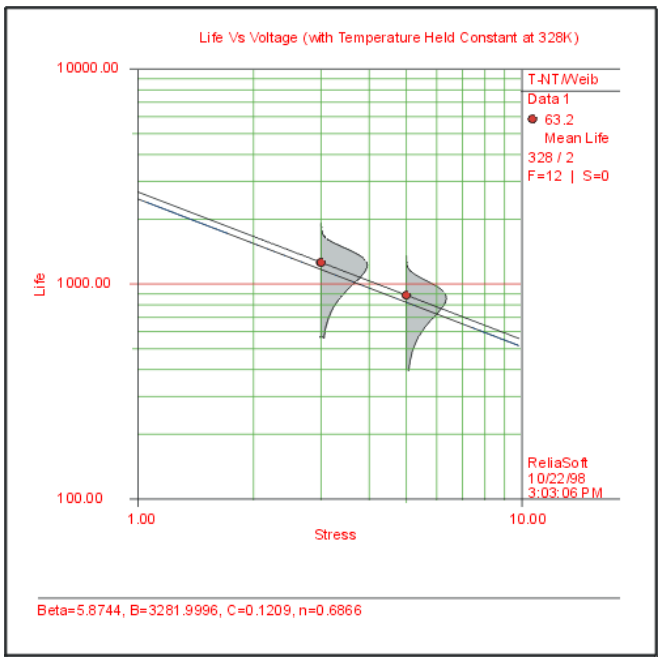


Figure 6.14: The effects of voltage on life, with temperature held constant.

Following figure shows the life vs. temperature plot with voltage held constant at 2V.

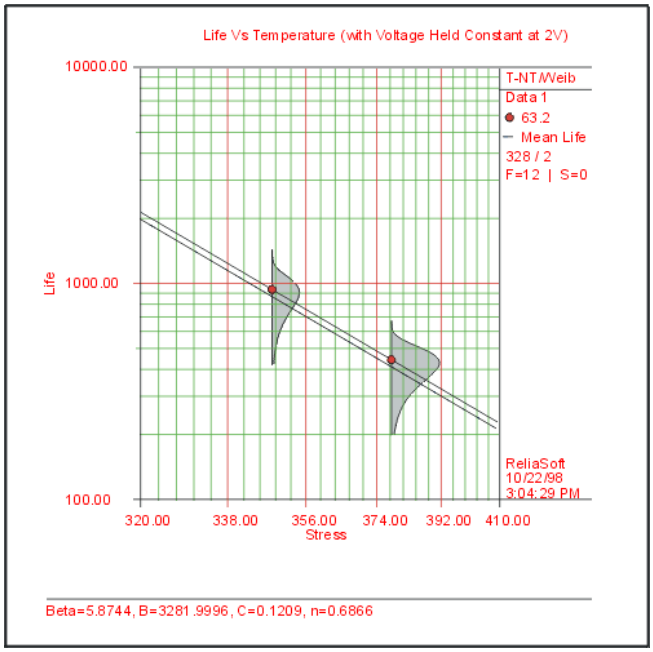


Figure 6.15: The effects of temperature on life, with voltage held constant.

ELECTRONIC COMPONENTS EXAMPLE

An electronic component was redesigned and was tested to failure at three different temperatures. Six units were tested at each stress level. At the 406K stress level however, a unit was removed from the test due to a test equipment failure, which led to a failure of the component. A warranty time of one year is to be given, with an expected return of 10% of the population. The times-to-failure and test temperatures are given next:

Time-to-Failure (hr)		
406K	**416K**	**426K**
(0.3)	164	92
248	176	105
456	289	155
528	319	184
731	340	219
813	543	235

The operating temperature is 356K. Using the Arrhenius-Weibull model, determine the following:

1. Should the first failure at 406K be included in the analysis?
2. Determine the warranty time for 90% reliability.

3. Determine the 90% lower confidence limit on the warranty time.
4. Is the warranty requirement met? If not, what steps should be taken?
5. Repeat the analysis with the unrelated failure included. Is there any difference?
6. If the unrelated failure occurred at 500 hr, should it be included in the analysis?

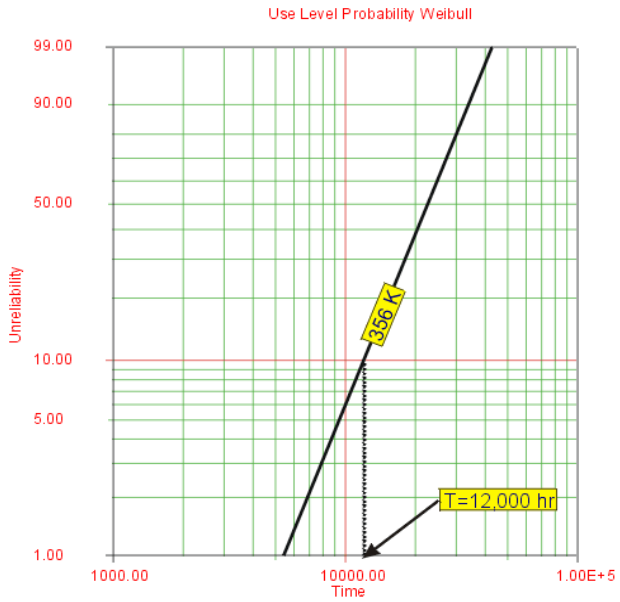
SOLUTION

1. Since the failure occurred at the very beginning of the test and for an unrelated reason, it can be omitted from the analysis. If it is included it should be treated as a suspension and not as a failure.

2. The first failure at 406K was neglected and the data were analyzed using ALTA. The following parameters were obtained:

$\beta = 2.9658,$
$B = 10679.57,$
$C = 2.39662 \cdot 10^{-9}$

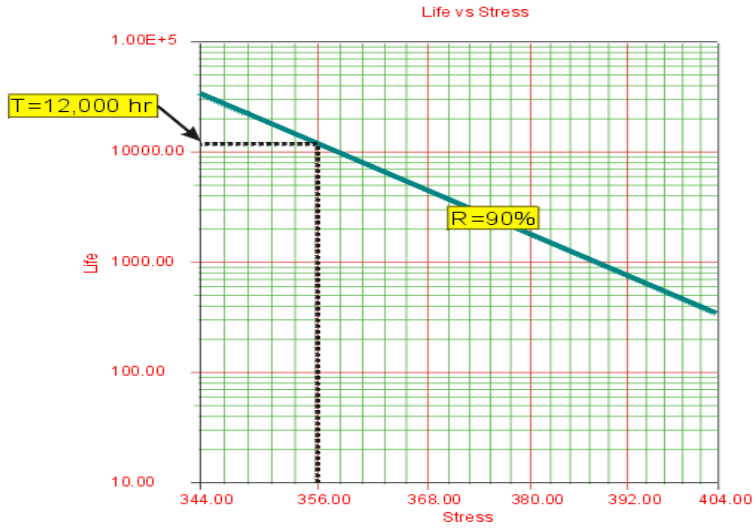
The use level probability plot (at 356K) can then be obtained. The warranty time for a reliability of 90% (or an unreliability of 10%) can be estimated from this plot as shown next.



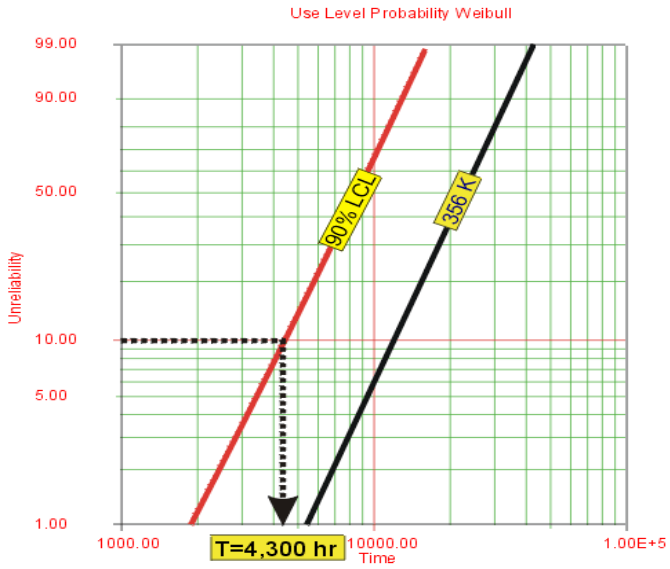
This estimate can also be obtained from the Arrhenius plot (a life vs. stress plot). The 10th percentile (time for a reliability of 90%) is plotted

versus stress. This type of plot is useful because a time for a given reliability can be determined for different stress levels.

A more accurate way to determine the warranty time would be to use ALTA's Quick Calculation Pad (QCP). By selecting the Warranty (Time) Information option from the Basic Calculations tab in the QCP and entering 356 for the temperature and 90 for the required reliability, a warranty time of 11,977.793 hr can be determined, as shown next:

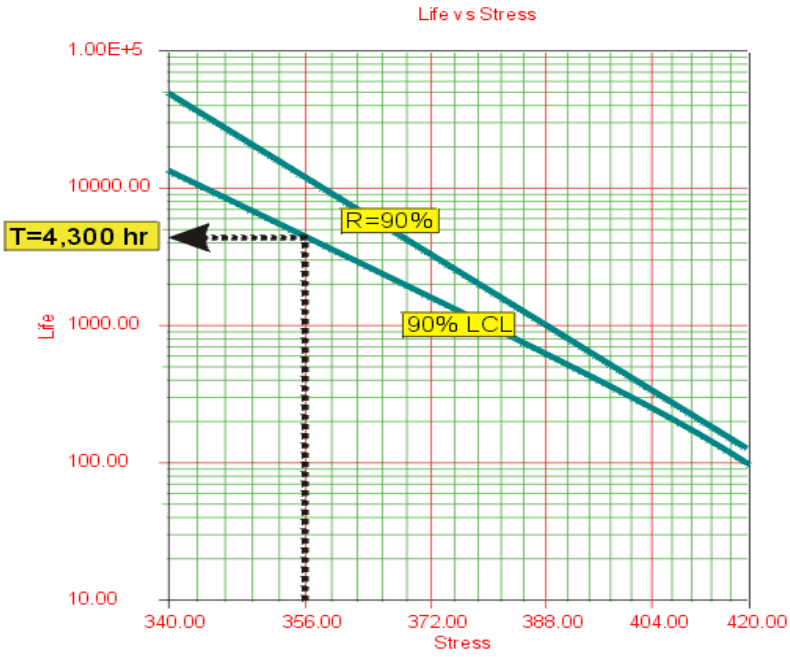


3. The warranty time for a 90% reliability was estimated to be approximately 12,000 hr. This is above the 1 year (8,760 hr) requirement. However, this is an estimate at the 50% confidence level. In other words, 50% of the time life will be greater than 12,000 hr and 50% of the time life will be less. A known confidence level is therefore crucial before any decisions are made. Using ALTA, confidence bounds can be plotted on both Probability and Arrhenius plots. In the following use level probability plot, the 90% Lower Confidence Level (LCL) is plotted. Note that percentile bounds are type 1 confidence bounds in ALTA.

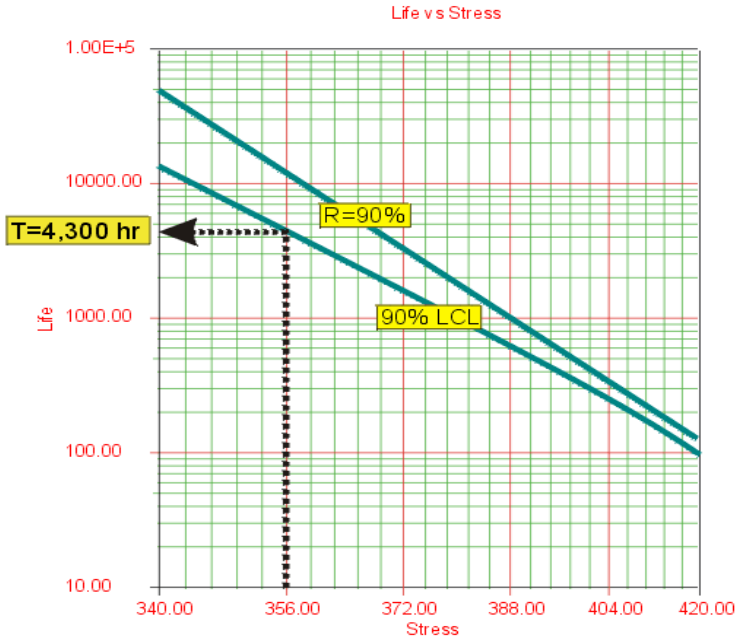


An estimated 4,300 hr warranty time at a 90% lower confidence level was obtained from the use level probability plot. This means that 90% of the time, life will be greater than this value. In other words, a life of 4,300 hr is a bounding value for the warranty.

The Arrhenius plot with the 90% lower confidence level is shown next.



Using the QCP and specifying a 90% lower confidence level, a warranty time of 4436.5 hr is estimated, as shown next.



4. The warranty time for this component is estimated to be 4,436.5 hr at a 90% lower confidence bound. This is much less than the 1 year warranty time required (almost 6 months). Thus the desired warranty is not met. In this case, the following four options are available:

   - redesign
   - reduce the confidence level
   - change the warranty policy
   - test additional units at stress levels closer to the use level

5. Including the unrelated failure of 0.3 hr at 406 K (by treating it as a suspension at that time), the following results are obtained:

$$\beta = 2.9658,$$
$$B = 10679.57$$
$$C = 2.39662 \cdot 10^{-9}$$

These results are identical to the ones with the unrelated failure excluded. A small difference can be seen only if more significant digits are considered. The warranty time with the 90% lower 1-sided confidence bound was estimated to be:

$$T = 11.977.729 \text{ hr},$$
$$T_L = 4436.46 \text{ hr}.$$

Again, the difference is negligible. This is due to the very early time at which this unrelated failure occurred.

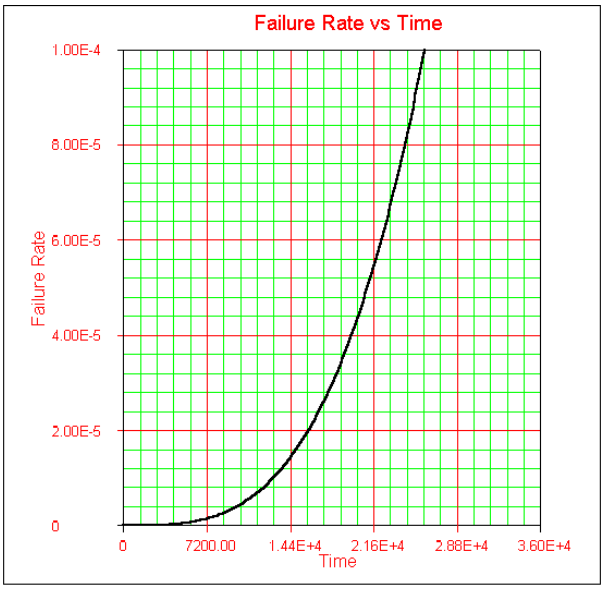
6. The analysis is repeated treating the unrelated failure at 500 hr as a suspension, with the following results:

$$\hat{\beta} = 3.0227,$$
$$B = 10959.52,$$
$$C = 1.23808 \cdot 10^{-9}$$

In this case, the results are very different. The warranty time with the 90% lower 1-sided confidence bound is estimated to be:

$$T = 13780.208 \text{ hr},$$
$$T_L = 5303.67 \text{ hr}.$$

It can be seen that in this case, it would be a mistake to neglect the unrelated failure. By neglecting this failure, we would actually underestimate the warranty time. The important observation in this example is that every piece of life information is crucial. In other words, unrelated failures also provide information about the life of the product. An unrelated failure occurring at 500 hr indicates that the product has survived for that period of time under the particular stress level, thus neglecting it would be a mistake. On the other hand it would also be a mistake to treat this data point as a failure, since the failure was caused by a test equipment failure.



7. Highly Accelerated Testing

7.1. Introduction

Highly Accelerated Life Tests (HALT) and Highly Accelerated Stress Screens (HASS) are briefly introduced and discussed in what follows. These techniques have been successfully used by many organizations for three decades. Most of these users do not publish their results because of the pronounced financial and technical advantages of the techniques over the classical methods, which are not even in the same league in terms of speed and cost. It is important to note that the methods are still rapidly evolving.

The HALT and HASS methods are designed to improve the reliability of the products, not to determine what the reliability is. The approach is therefore proactive as compared with a Reliability Demonstration (Rel-Demo) or Mean time between failures (MTBF) tests that do not improve the product at all but simply (attempt to) measure what the reliability is. This is a major difference between the classical and the HALT approaches.

7.2. Why Things Fail?

A product will fail when the applied load exceeds the strength of the product. The load can be voltage, current, force, temperature or other variable. Consider applied load and strength plotted together as in Figure 7.1.

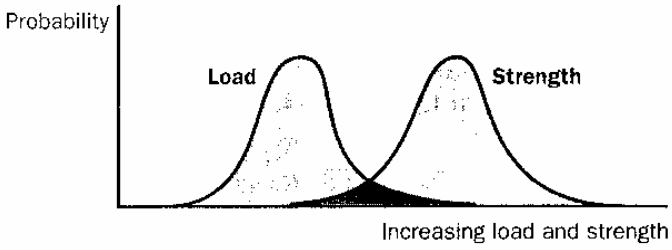


Figure 7.1: Load and strength

Whenever the applied load exceeds the strength, failure will occur. The load may be a one time load or it may be applied a number of times. In the first case, overload failure will occur and in the second case fatigue failure will occur. A fatigue

could be drawn for either case and would look similar to that shown. The crosshatched area represents the products which will fail.

As time passes, the product could become weaker for any one of many reasons. Figure 7.2 is concerned with aging. Alternatively, one could depict fatigue damage by having the strength curve move to the left as depicted in Figure 7.2. Again, when the applied load exceeds the strength, failure will occur. Either way, the overlap of the curves will increase, meaning that more products will fail. This moving of the curve can also be simulated by moving the applied load curve to the right as depicted in Figure 7.3. Note that one would have the same failures as when the strength degraded. It is this last approach that is taken in HALT, wherein the loads are increased until failure occurs, identifying a weakness. It is seen that one would obtain the same failures in either case according to the illustration. This simplistic example is quite valid and one could go through detailed calculations to demonstrate the fact. It will be left as a simple illustration here.

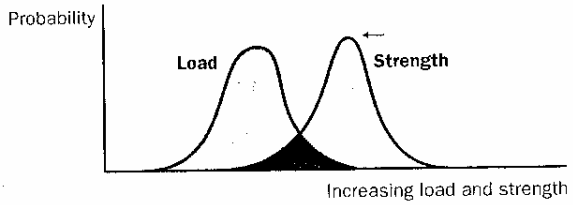


Figure 7.2: Load and increasing strength

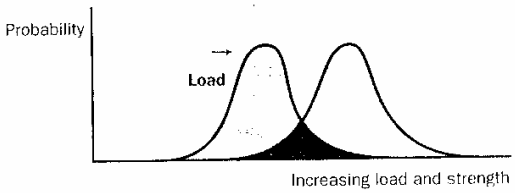


Figure 7.3: Strength and increasing load

7.2.1. The Bathtub Curve

The pattern of failures that occurs in the field can be approximated in three ways. When there are defects in the product, so-called "infant mortalities", or failure of weak items, items, will occur. Another type of failure is due to externally induced failures where load exceeds strength. Finally, wearout will occur even if an item is not defective. When one superimposes all three types of failure, a curve called the bathtub curve occurs. One such curve is shown in Figure 7.5. The bathtub curve is grossly affected by HALT and HASS techniques:

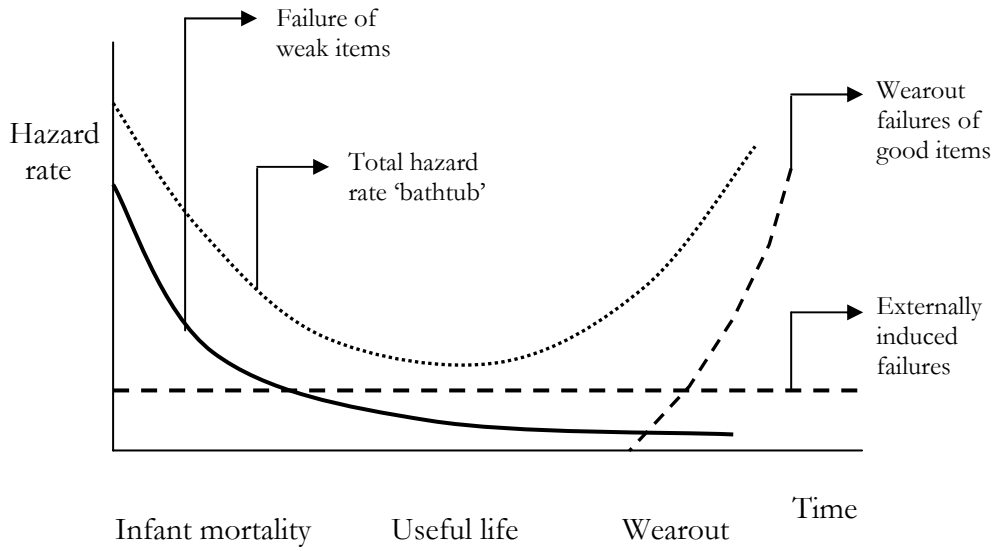


Figure 7.4: The bathtub curve

1. Production screening (HASS), will reduce the early segment of the curve by eliminating early life failures due to manufacturing flaws.
2. Ruggedization (HALT) of the product will lower the mid-portion of the curve which is due to externally induced failures.
3. HALT will extend the wearout segment far to the right.

7.3. The Purposes of HALT and HASS

The general purposes of applying accelerated stress conditions in the design phases is to find and improve upon design and process weaknesses in the least amount of time and correct the source of the weaknesses before production beings. It is generally true that robust products will exhibit much higher reliability than non-robust ones and so the ruggedization process of HALT in which large margins are obtained will generate products of high potential reliability. In order to achieve the potential, defect-free hardware must be manufactured or, at least, the defects must be found and fixed before shipment. In HASS, accelerated stresses are applied in production in order to shorten the time to failure of the defective units and therefore shorten the corrective action time and the number of units built with the same flaw. Each weakness found in HALT or in HASS represents an opportunity for improvement. The application of accelerated stressing techniques to force rapid design maturity (HALT) results in pay-backs that far exceed these from production stressing (HASS). Nonetheless, production HASS is cost effective in its own right until quality is such

that a sample HASS or Highly Accelerated Stress Audit (HASA) can be put into place. The use of HASA demands excellent process control since most units will be shipped without the benefit of HASS being performed on them, and only those units in the selected sample will be screened for defects.

The stresses used in HALT and HASS include, but are not restricted to, all-axis simultaneous vibration, high-rate broad-range temperature cycling, power cycling, voltage and frequency variation, humidity, and any other stress that may expose design or process problems. No attempt is made to simulate the field environment. One only seeks to find design and process flaws by any means possible. The stresses used generally far exceed the field environments in order to gain time compression; that is, shorten the time required to find any problems areas. When a weakness is discovered, only the failure mode and mechanism is of importance, the relation of the stress used to the field environment is of no consequence at all. Figure 7.5 illustrates this point. In this figure, λ is the instantaneous failure rate for a given failure mode. The two curves illustrating a thermally induced failure rate and a vibration-induced failure rate are located so that the field stresses at which failure occurs and the HALT stresses at which failure occurs are lined up vertically. It is then seen that a failure mode that would most often be exposed by temperature will be more likely to be exposed by vibration in the HALT environment.

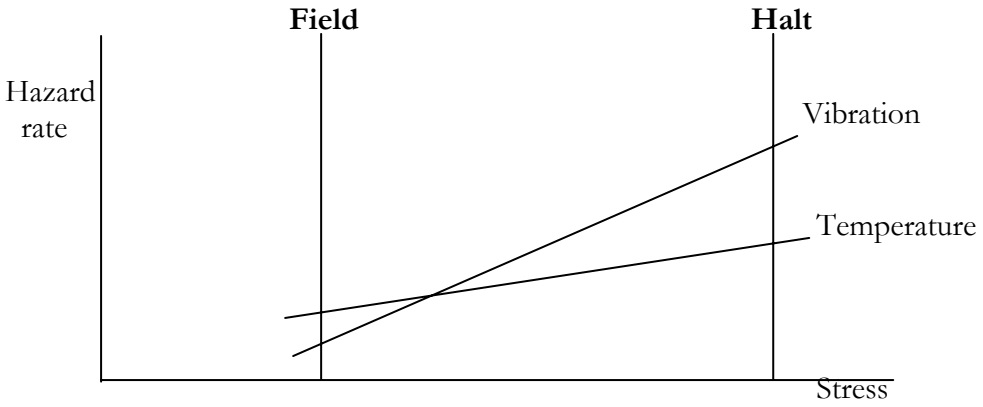


Figure 7.5: Instantaneous failure rates in the field and in HALT

It is very common to expose weaknesses in HALT with a different stress than the one that would make the weakness show up in the field. It is for this reason that one should focus on the failure mode mechanism instead of the margin for the particular stress in use.

"Mechanism" here means the conditions that caused the failure, such as melting, exceeding the stable load or exceeding the ultimate strength. The corresponding failure mode could be separation of a conductor, elastic buckling and tensile failure, respectively. Considering the margin instead of the failure mode is a major mistake which is made by most engineers used to conventional test techniques. In HALT and HASS, one uses extreme stresses for a very brief period of time in order to obtain time compression in the failures. In doing so, one may obtain the same failures as would occur in the field environments, but with a different stress. For example, a water sprinkler manufacturer had a weakness which was exposed by the diurnal thermal cycle in the field. HALT exposed the same weakness with all-axis vibration after extensive thermal cycling failed to expose the weakness. After the weakness was addressed, the field failures were eliminated, which proves that the weakness exposed by all-axis vibration was a valid discovery. For another example, consider a reduction in the cross-sectional area of a conductor. This reduction would create a mechanical stress concentration and an electrical current density concentration. This flaw might be exposed by temperature cycling or vibration in HALT or HASS and might be exposed by electromigration in the field environment. Either way, the reduction in area introduces a weakness that can be eliminated.

In addition to stresses, other parameters are used to look for weaknesses. These include the diameter of a gear, the pH of a fluid running through the product, contaminants in the fluid running through a blood analyzer, the thickness of a tape media, the viscosity of a lubricant, the size of a tube or pipe, the lateral load on a bearing and an almost endless additional number of factors. What is sought is any information that could lead to an opportunity for improvement by decreasing the sensitivity of the product to any conditions which could lead to improper performance or to catastrophic failure. Anything that could provide information for an improvement in the margin is appropriate in HALT.

In the HALT phase of product development, which should be in the early design phase, the product is improved in every way practicable bearing in mind that most of what are discovered in HALT as weaknesses will almost surely become field failures if not improved. This has been demonstrated thousands of times by users of HALT. Of course, one must always use reason in determining whether or not to improve the product when an opportunity is found and this is done by examining the

failure mode and mechanism. Just because a weakness was found "out of specification" is no reason to reject the finding as an opportunity for improvement. There are numerous cases where weaknesses found "out of specification" were not addressed until field failures of the exact same type occurred. If you find it in HALT, it is probably relevant. In various papers from Hewlett-Packard over the years, it has been found that most of the weaknesses found in HALT and not addressed resulted in costs to the company in the neighborhood of US$10,000,000 per failure mode to address later, when failure costs were included. It cannot be emphasized too much that it is imperative to focus on the failure mode and mechanism and not the conditions used to make the weakness apparent. Focusing on the margin will usually lead one to allow a detected weakness to remain, resulting in many field failures of that type before a fix can be implemented. Learn from others' mistakes and do not focus on the stress level used, but on the failure mode and mechanism.

HALT and HASS are not restricted to electronic boxes, but apply to many other technologies as well. Some of the technologies are listed at the end of the chapter and include such diverse products as shock absorbers, airframes, auto bodies, exhaust systems and power steering hoses to name just a few. Note that HALT addresses design and process weaknesses, whereas classical ESS only addresses production weaknesses. HASS may expose design weaknesses if any remain or are introduced after production start.

7.4. Equipments Required

The application of the highly accelerated stress techniques is very much enhanced by, if not impossible without, the use of environmental equipment of the latest design such as all-axis exciters and combined very high-rate thermal chambers (60 °C/min or more *product* rate). All-axis means three translations and three rotations.

A single-axis, single-frequency shaker will only excite modes in the particular direction of the vibration and only those nearby in frequency. A swept sine will sequentially excite all modes in the one direction being excited. A single-axis random shaker will simultaneously excite all modes in one direction. A six-axis system will simultaneously excite all modes within the bandwidth of the shaker in all directions. If all modes in all directions are not excited simultaneously, then many defects can be

missed. Obviously, the all-axis shakers are superior for HALT and HASS activities since one is interested in finding as much as possible as fast as possible.

In the very early days of Design Ruggedization (the precursor to HALT), a device had been severely ruggedized using a single-axis random shaker system. Then, in production, a very early all-axis system was used and three design weaknesses which had not been found on the single-axis system were exposed almost immediately. That experience showed the differences in the effectiveness of the various systems. Since then, the system of choice has been an all-axis broad-band shaker.

Other types of stresses or other parameters may be used in HALT. In these cases, other types of stressing equipment may be required. If one wanted to investigate the capability of a gearbox, one could use contaminated oil, out-of-specification gear sizes and a means for loading the gearbox in torsion either statically or dynamically. If one wanted to investigate various end piece crimping designs on power steering hoses, one could use temperature, vibration and oil pressure simultaneously. This has been done and worked extremely well, exposing poor designs in just a few minutes. In order to investigate an airframe for robustness in pressurization, the hull could be filled with water and rapid pressure cycling done. This is show it is done at several aircraft manufacturers. Water is used as the pressurized medium since it is nearly incompressible and so when a fracture occurs, pressure drops quickly, preventing an explosive-type failure, such as would occur if air were to be used. A life test simulating thousands of cycles can be run in just a few days using this approach.

Not that, in HALT and HASS, one tries to do fatigue damage as fast as possible, and the more rapidly it is done, the sooner it can stop and the less equipment is needed to do the job. It is not unusual to reduce equipment is needed to do the job. It is not unusual to reduce equipment costs by *orders* of magnitude by using the correct stresses and accelerated techniques. This comment applies to all environmental stimulation and not just to vibration. An example discussed later in this book (Chapter 7) shows a decrease in cost from US$22 million to US$50,000 on thermal chambers alone (not counting power requirement, associated vibration equipment, monitoring equipment and personnel) by simply increasing the rate of

change of temperature from 5 °C/min to 40 °C/min (when rate-sensitive flaws are present)! The basic data for this comparison is given in [11]. Another example shows that increasing the RMS vibration level by a factor of 1.4 times would decrease the vibration system cost from US$100 million to only US$100,000 for the same throughout of product. With these examples, it becomes clear that HALT and HASS techniques, when combined with modern screening equipment designed specifically to do HALT and HASS, provide quantum leaps in cost effectiveness.

Some typical results of HALT and HASS applied to product design and manufacturing are described below. Some of these are from early successes and have been published in some form, usually technical presentations at a company. Later examples using the alter technology in terms of technique and equipment have largely not been published. The later results are, of course, much better, but the early results will make the point well enough, since they represent a lower bound on the expected successes today when far better techniques are equipment are available.

7.5. Some General Comments on HALT and HASS

The successful use of HALT or HASS requires several actions to be completed. In sequence these are: precipitation, detection, failure analysis, corrective action, verification of corrective action and then entry into a database. *All* of the first five must be done in order for the method to function at all. Adding the sixth results in long-term improvement of the future products.

1. *Precipitation* means to change a defect which is latent or undetectable to one that is patent or detectable. A poor solder joint is such an example. When latent, it is probably not detectable electrically unless it is extremely poor. The process of precipitation will transpose the flaw to one that is detectable; that is, cracked. This cracked joint may be detectable under certain conditions, such as modulated excitation. The stresses used for the transformation may be vibration combined with thermal cycling and perhaps electrical overstress. Precipitation is usually accomplished in HALT or in a precipitation screen.
2. *Detection* means to determine that a fault exists. After precipitation by whatever means, it may become patent that is, detectable. Just because it is patent does not mean that it will actually be detected since it must first be put into a detectable state, perhaps using modulated excitation, and then it must

actually be detected. Assuming that we actually put the fault into a detectable state and that the built-in test or external test setup can detect the fault, we can then proceed to the most difficult step, which is failure analysis.

3. *Failure analysis* means to determine why the failure occurred. In the case of the solder joint, we need to determine why the joint failed. If doing HALT, the failed joint could be due to a design flaw; that is, an extreme stress at the joint due to vibration or maybe due to a poor match of thermal expansion coefficients. When doing HASS, the design is assumed to be satisfactory (which may not be true if changes have occurred) and, in that case, the solder joint was probably defective. In what manner it was defective and why it was defective. In what manner it was defective and why it was defective need to be determined in sufficient detail to perform the next step, which is corrective action.

4. *Corrective action* means to change the design or processes as appropriate so that the failure will not occur in the future. This step is absolutely essential if success is to be accomplished. In fact, corrective action is the main purpose of performing HALT or HASS.

5. *Verification of corrective action* needs to be accomplished by testing to determine that the product is really fixed and that the flaw which caused the problem is no longer present. The fix could be ineffective or there could be other problems causing the anomaly which are not yet fixed. Additionally, another fault could be induced by operations on the product and this necessitates a repeat of the conditions that promoted the fault to be evident. Note that a test under zero stress conditions will usually not expose the fault. One method of testing a fix during the HALT stage is to perform HALT again and determine that the product is at least as robust as it was before and it should be somewhat better. If one is in the HASS stage, then performing HASS again on the product is in order. If the flaw is correctly fixed, then the same failure should not occur again.

It is essential to have at least the first five steps completed in order to be successful in improving the reliability of a product. If any one of the first five steps is not completed correctly, then no improvement will occur and the general trend in reliability will be toward a continuously lower level.

6. The last step is to put the lesson learned into a database from which one can extract valuable knowledge whenever a similar event occurs again. Companies which practice correct HALT and utilize a well-kept database soon become very adept at designing and building very robust products with the commensurate high reliability. These companies usually are also very accomplished at HASS and so can progress to HASA, the audit version of HASS.

A comparison of the HALT and HASS approach and the classical approach is presented in Table 7.1. Note that HALT and HASS are proactive, i.e. seek to improve the product's reliability, and much of the classical approaches are intended to measure the product's reliability, not to improve it.

Table 7.1: Comparison of HALT and Classical Approaches

Stage	**Design**			**Pre-production**				**Production**
Test type	Quality	HALT	Life test	HASS development	Safety of HASS	Rel-Demo	HASS	HASS
Purpose	Satisfy Customer Reqmts	Maximize margins, minimize sample	Demo life	Select screens and equip	Prove Ok to ship	Measure reliability	Improve reliability	Minimize cost, maximize effectiveness
Desired outcome	Customer acceptance	Improve margins	MTBF and spares reqd	Minimize cost, maximize HASS reliability	Life left after	Pass	Root cause corrective action	Minimize cost, maximize effectiveness
Method	Simulate field environment sequentially	Step stress to failure	Simulate Field	Maximize time compression	Multiple repeats without wearout	Simulate field	Accelerated stimulation	Repeat HASS, Modify profits
Duration Stress field level	Weeks Field	Days Exceeds field	Weeks Field	Days Exceeds field	Days Exceeds field	Months Field	Minutes Exceeds	Weeks Exceeds field

8. Accelerated Life Testing Concepts and Models

Product reliability contributes much to quality and competitiveness. Many manufacturers yearly spend millions of dollars on product reliability. Much management and engineering effort goes into evaluating reliability, assessing new designs and design and manufacturing changes, identifying causes of failure, and comparing designs, vendors, materials, manufacturing methods, and the like. Major decisions are based on life test data, often from a few units. Moreover, many products last so long that life testing at design conditions is impractical. Many products can be life tested at high stress conditions to yield failures quickly. Analyses of data from such an accelerated test yield needed information on product life at design conditions (low stress).

Many of today's applications demand that the products must be capable of operating under extremes of environmental stress and for thousands of hours without failure. For such demanding situations, the traditional tests are no longer sufficient to identify design weaknesses or validate life predictions.

Accelerating testing is an approach for obtaining more information from a given test and time that would be impossible under normal circumstances. We do this by using a test environment that is more severe than that is experienced in normal use conditions with a rider to avoid introducing failure modes that would not be encountered in normal use.

What is Accelerated Life Testing?

Traditional life data analysis involves analyzing times-to-failure data (of a product, system or component) obtained under normal operating conditions in order to quantify the life characteristics of the product, system or component. In many situations, and for many reasons, such life data (or times-to-failure data) is very difficult, if not impossible, to obtain. The reasons for this difficulty can include the long life times of today's products, the small time period between design and release and the challenge of testing products that are used continuously under normal conditions. Given this difficulty, and the need to observe failures of products to better understand their failure modes and their life characteristics, reliability

practitioners have attempted to devise methods to force these products to fail more quickly than they would under normal use conditions. In other words, they have attempted to accelerate their failures. Over the years, the term *accelerated life testing* has been used to describe all such practices.

Accelerated testing: Briefly stated, accelerated testing consists of variety of test methods for shortening the life of products or hastening the degradation of their performance. The aim of such testing is to quickly obtain data which, properly modeled and analyzed, yield desired information on product life or performance under normal use. Obviously, such testing saves much time and money.

8.1. Test Purpose

Accelerated life tests and performance degradation test serve various purposes. Common purposes include:

1. Identify design failures. Eliminate or reduce them through redundancy, better design, components, etc.
2. Comparisons. Choose among designs, components, suppliers, rated operating conditions, test procedures, etc.
3. Identify manufacturing defects. Eliminate them through better manufacturing, components, burn-in, etc. Estimate the reliability improvement from eliminating or reducing certain failure modes.
4. Evaluate other variables. Assess how much design, manufacturing, materials, operating, and other variables affect reliability. Optimize reliability with respect to them. Decide which need to be controlled. Measure reliability. Assess whether to release a design to manufacturing or product to a customer. Estimate warranty and service costs, failure rates, mean time to failure (MTTF), degradation rates, etc. Satisfy a customer requirement for such measurement. Use as marketing information.
5. Demonstrate reliability. Show that product reliability surpasses customer specifications.
6. Operating conditions. Develop relationships between reliability (or degradation) and operating conditions. Choose design operating conditions.

7. Service policy. Decide when to inspect, service, or replace and how many spares and replacements to manufacture and stock. Units may be taken out of service and tested under accelerated conditions when an unexpected problem shows up in service

The applications of ALT and benefits drawn have no bounds. Look at the areas where this technology has been successfully applied:

8.1.1. On Materials

Metal: Accelerated testing is used with metals, including test coupons and actual parts, as well as composites, welds, brazements, bonds, and other joints. Performance includes fatigue life, creep, creep-rupture, crack initiation and propagation, wear, corrosion, oxidation, and rusting. Accelerating stresses include mechanical stress, temperature, specimen geometry and surface finish. Chemical acceleration factors include humidity, salt, corrosives, and acids.

Plastics: Accelerated testing is used with many plastics including building materials, insulation (electrical and thermal), mechanical components, and coatings. Materials include polymers, polyvinyl chloride (PVC), urethane foams, and polyesters. Performance includes fatigue life, wear, mechanical properties, and color fastness. Accelerating stresses include mechanical load (including vibration and shock), temperature (including cycling and shock), and weathering (ultraviolet radiation and humidity).

Dielectrics and insulations: Accelerated testing is used with many dielectrics and electrical insulations including solids (polyethylene, epoxy), liquids (transformer oil), gases, and composites (oil-paper, epoxy-mica). Products include capacitors, cables, transformers, motors, generators, and other electrical apparatus. Performance includes time to failure and other properties (breakdown voltage, elongation, ultimate mechanical strength). Accelerating stresses include temperature, voltage stress, thermal and electrical cycling and shock, vibration, mechanical stress, radiation and moisture.

Ceramics: Applications are concerned with fatigue life, wear, and degradation of mechanical and electrical properties.

Adhesives: Accelerated testing is used with adhesive and bonding materials such as epoxies. Performance includes life and strength. Accelerating stresses include mechanical stress, cycling rate, mode of loading, humidity, and temperature.

Rubber and elastics: Accelerated testing is used with rubbers and elastic materials (e.g., polymers). Products include tires and industrial belts. Performance includes fatigue life and wear. Accelerating stresses include mechanical load, temperature, pavement texture, and weathering (solar radiation, humidity, and ozone).

Food and drugs: Accelerated testing is used with foods (e.g., browning of white wines), drugs, pharmaceuticals, and many other chemicals. Performance is usually shelf (or storage) life, usually in terms of amount of an active ingredient that degrades. Performance variables include taste, pH, moisture loss or gain, microbial growth, color, and specific chemical reactions. Accelerating variable include temperature, humidity, chemicals, pH, oxygen, and solar radiation.

Lubricants: Accelerated testing is used with solid (graphite, molybdenum disulphide, and teflon), oil, grease, and other lubricants. Performance includes oxidation, evaporation, and contamination. Accelerating stresses include speed, temperature, and contaminants (water, copper, steel, and dirt).

Protective coatings and paints: Accelerated testing is used for weathering of paints (liquid and powder), polymers, antioxidants, anodized aluminum, and electroplating. Performance includes color, gloss, and physical integrity (e.g., wear, cracking, and blistering). Accelerating stresses include weathering variables-temperature, humidity, solar radiation (wavelength and intensity) – and mechanical load.

Concrete and cement: Accelerated testing is used with concrete and cement to predict performance-the strength after 28 days of curing. The accelerating stress is high temperature applied for a few hours.

Building materials: Accelerated testing is used with wood, particle board, plastics, composites, glass, and other building materials. Performance includes abrasion résistance, color fastness, strength, and other mechanical properties. Accelerating stresses include load and weathering (solar radiation temperature, humidity).

Nuclear rector materials: Accelerated testing is used with nuclear rector materials, for example, fuel rod cladding. Performance includes strength, creep and creep-rupture. Accelerating stresses include temperature, mechanical stress, contaminants, and nuclear radiation (type, energy, and flux).

8.1.2. On Products

Semiconductors and microelectronics. Accelerated testing is used for many types of semiconductor devices including transistors such as gallium arsenide field emission transistors (GaAs FETs), insulated gate field emission transistors (IGFETs), Gunn and light emitting diodes (LEDs), MOS and CMOS devices, random access memories (RAMs), and their bonds, connections, and plastic encapsulants. They are tested singly and in assemblies such as circuit boards, integrated circuits (LSI and VLSI), and microcircuits. Performance is life and certain operating characteristics. Accelerating variables include temperature (constant, cycled, and shock), current, voltage (bias), power, vibration and mechanical shock, humidity, pressure, and nuclear radiation.

Capacitors. Accelerated testing is used with most types of capacitors, including electrolytic, polypropylene, thin film, and tantalum capacitors. Performance is usually life. Accelerating variables include temperature, voltage, and vibration.

Resistors. Accelerated testing is used with thin and thick film, metal oxide, pyrolytic, and carbon film resistors. Performance is life. Accelerating variables include temperature, current, voltage, power vibration, electrochemical attack (humidity), and nuclear radiation.

Other electronics. Accelerated testing is used with other electronic components such as optoelectronics (opto couplers and photo conductive cells), lasers, liquid crystal displays, and electric bonds and connections.

Electrical contacts. Accelerated testing is used for electrical contacts in switches, circuit breakers, and relays. Performance includes corrosion and life. Metal fatigue, rupture, and welding are common failure mechanisms. Accelerating stresses include high cycling rate, temperature, contaminants (humidity), and current.

Cells and batteries. Accelerated testing is used with rechargeable, non-rechargeable, and solar cells. Performance includes life, self discharge, current, and depth of discharge. Accelerating variables include temperature, current density, and rate of charge and discharge.

Lamps. Accelerated testing is used with incandescent (filament), fluorescent (including ballasts), mercury vapor, and flash lamps. Performance includes life, efficiency, and light output. Accelerating variables include voltage, temperature, vibration, and mechanical and electrical shock.

Electrical devices. Accelerated testing is used with various electrical devices including motors, heating elements, and thermoelectric converters.

Bearings. Accelerated testing is used with roller, ball, and sliding (oil film) bearings. Performance includes life and wear (weight loss). Materials include steels and silicon nitride for rolling bearings and porous (sintered) metals, bronzes, babbitt, aluminum alloys, and plastics for sliding bearings. Accelerated stresses include overspeed, mechanical load, and contaminants.

Mechanical components. Accelerated testing is used with mechanical components and assemblies such as automobile parts, hydraulic components, tools, and gears. Performance includes life and wear. Accelerating stresses include mechanical load, vibration, temperature and other environmental factors, and combinations of such stresses.

In the above application areas, the accelerating factors used, either singly or in combinations, which include:

- More frequent power cycling
- Higher vibration levels
- High humidity
- More severe thermal cycling
- Higher temperatures
- Mechanical load

Accelerating testing is a powerful tool that can be effectively used in two very different ways, viz., in a qualitative or in a quantitative manner. Therefore, the Accelerated testing can be divided into two broad areas, viz.,

- *Qualitative Accelerated Testing-problem/weakness identification and correction*

  Where the concern is directed towards identifying failures and failure modes without attempting to make any predictions as to the product's life under normal use conditions. In qualitative accelerated testing, the engineer is mostly interested in identifying failures and failure modes without attempting to make any predictions as to the product's life under normal use conditions. Qualitative tests are performed on small samples with the specimens subjected to a single severe level of stress, to a number of stresses or to a time-varying stress (*i.e.* stress cycling, cold to hot, etc.). If the specimen survives, it passes the test. Otherwise, appropriate actions will be taken to improve the product's design in order to eliminate the cause(s) of failure. *Qualitative tests are used primarily to reveal probable failure modes. However, if not designed properly, they may cause the product to fail due to modes that would have never been encountered in real life.* A good qualitative test is one that quickly reveals those failure modes that will occur during the life of the product under normal use conditions. In general, qualitative tests are not designed to yield life data that can be used in subsequent quantitative accelerated life data analysis as described in this reference. In general, qualitative tests do not quantify the life (or reliability) characteristics of the product under normal use conditions, however, they provide valuable information as to the types and level of stresses one may wish to employ during a subsequent quantitative test. Obviously, these test can not provide the answer to the question-What will be the reliability of the product under normal use conditions?

- *Quantitative Accelerated Life Testing-Life estimation.*

  The engineer is concerned towards predicting the life of the product (or more specifically, life characteristics such as MTTF, B_{10} life, etc.) at normal use conditions, from data obtained in an accelerated life test. It consists of tests designed to quantify the life characteristics of the product, component or system under normal use conditions and thereby provide reliability

information. Reliability information can include the determination of the probability of failure of the product under use conditions, mean life under use conditions and projected returns and warranty costs. It can also be used to assist in the performance of risk assessments, design comparisons, etc.

In accelerated testing the quantitative knowledge builds upon the qualitative knowledge. In fact, the accelerated testing in a quantitative manner requires a physics-of-failure approach, i.e., a comprehensive understanding and application of specific failure mechanism involved and the relevant stress(es). Table 8.1 compares the two main categories of ALT.

Some accelerating techniques are appropriate only for part level whereas others could be used for higher levels of assembly. Very few techniques could be applicable to both part and assembly, where the underlying assumptions and modeling may be valid at the part level may be totally invalid tests performed at higher level of assembly or vise versa.

Table 8.1: Types of ALT

Test	Purpose and approach	Comment
Qualitative	Uses accelerated environmental stresses to precipitate latent defects or design weaknesses into actual failures to identify design part or manufacturing process problems, which could cause subsequent failures in the field.	Requires a thorough understanding or at least a workable knowledge of the basic failure mechanism. Estimation of item life may or may not be of a concern.
Quantitative	Uses model relating the reliability (or life) measured under high stress conditions to that which is expected under normal operation.	Requires: An understanding of the anticipated failure(s) mechanism A knowledge of the magnitude of the acceleration of this failure mechanism as a function of accelerating stress

Unfortunately, there is no single magic analytical model that can accurately estimate the life of complex assemblies or system. Each life analytical model describes physical change mechanisms associated with specific material characteristics.

8.2. Types of Acceleration and Stress Loading

Two methods of acceleration, viz., *usage rate acceleration* and *overstress acceleration*, have been devised to obtain times-to-failure data at an accelerated pace. For products that do not operate continuously, one can accelerate the time it takes to induce failures by continuously testing these products. This is called usage rate acceleration. For

products for which usage rate acceleration is impractical, one can apply stress(es) at levels which exceed the levels that a product will encounter under normal use conditions and use the times-to-failure data obtained in this manner to extrapolate to use conditions. This is called overstress acceleration

High Usage Rate: A simple way to accelerate the life of many products is to run the product more-at a higher usage rate. The following are two common ways of doing such compressed time testing.

Faster: One way to accelerate is to run the product faster. For example, in many life tests, rolling bearings run at about three times their normal speed. High usage rate may also be used in combination with overstress testing. For example, such bearings are also tested under higher than normal mechanical load. Another example of high usage rate involves a voltage endurance test of an electrical insulation. The AC voltage in the test was cycled at 412 Hz instead of the normal 60 Hz, and test was shorted by a factor of $412/60 = 6.87$.

Reduced off time: Many products are off much of the time in actual use. Such products can be accelerated by running them a greater fraction of the time. For example, in most homes, a major appliance (say, washer or dryer) runs an hour or two a day; on test it runs 24 hours a day. In use, a refrigerator compressor runs about 15 hours a day; on test it runs 24. A small appliance (say, toaster or coffee maker) runs a few cycles a day; on test it cycles many times a day.

Purpose. The purpose such testing is to estimate the product life distribution at normal usage rates. It is assumed that the number of cycles, revolutions, hours, etc., to failure on test is the same that would be observed at the normal usage rate. For example, it is assumed that a bearing that runs 6.2 million revolutions to failure at high rpm would run 6.2 million revolutions at normal rpm. The data are treated as a sample from actual use. Then standard life data analyses provide estimates of the percentage failing on warranty, the median life, etc. They also provide comparisons of designs, manufacturing methods, materials, vendors, etc.

The assumption. It is not automatically true that the number of cycles to failure at high and normal usage rates is the same. Usually the test must be run with special care to assure that product operation and stress remain normal in all regards except

usage rate. For example, high rate usage usually raises the temperature of the product. That usually results in fewer cycles to failure. It may event produce failure modes not seen at normal temperature and usage rate. Thus many such tests involve cooling the product to keep the temperature at a normal level. In contrast, products sensitive to thermal cycling may last longer if run continuously without thermal cycling. For this reason, toasters on test are force cooled by a fan between cycles.

The limitation of usage rate acceleration arises when products, such as computer servers and peripherals, maintain a very high or even continuous usage. In such cases, usage acceleration, even though desirable, is not a feasible alternative. In these cases the practitioner must stimulate, usually through the application of stress(es), the product to fail. This method of accelerated life testing is called overstress acceleration and is described next.

8.2.1. Overstress Testing

Overstress testing consists of running a product at higher than normal levels of some accelerated stress(es) to shorten product life or to degrade product performance faster. Typical accelerating stresses are temperature, voltage, mechanical load, thermal cycling, humidity, and vibration or combination of these stresses. Overstress testing is the most common form of accelerated testing.

8.2.1.1. About Degradation Mechanisms

Fatigue. Materials eventually fail by fatigue if subjected to repeated mechanical loading and unloading, including vibration. Well studied are the fatigue of metals, plastics, glass, ceramics, and other structural and mechanical materials (see references on these). Fatigue is a major failure mechanism of mechanical parts including bearings and electrical contacts. The usual accelerating stress is load. Other stresses are temperature and chemicals (water, hydrogen, oxygen, etc.)

Creep. Creep, the slow plastic deformation of materials under constant mechanical load, may interfere with product function or cause rupture or fracture. Accelerating variables are typically temperature and mechanical load, load cycling, and chemical contaminants (for example, water, hydrogen, and fluorine).

Cracking. Metals, plastics, glass, ceramics, and other materials crack. People study crack initiation and growth. Accelerating stresses include mechanical stress, temperature, and chemicals (humidity, hydrogen, alkalis, and acids).

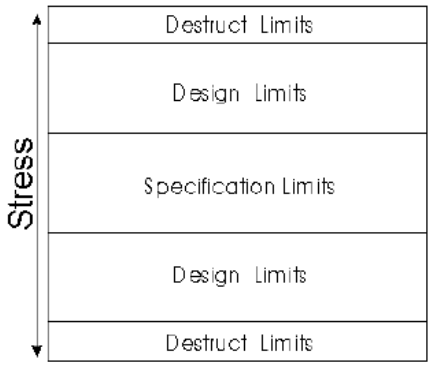
Wear. In applications, many materials are subjected to friction that removes the material. For example, rubber tires lose tread, house paints wash off, gears, bearings, and machine tools wear away. Accelerating stresses include speed, load (magnitude and type), temperature, lubrication, and chemicals (humidity).

Corrosion/oxidation. Most metals and many foods, pharmaceuticals, etc., deteriorate by chemically reacting with oxygen (oxidation and rusting), fluorine, chlorine, sulphur, acids, alkalis, salt, hydrogen peroxide, and water. Accelerating stresses include concentration of the chemical, activators, temperature, voltage, and mechanical load (stress-corrosion).

Weathering. This concerns the effects of weather on materials in outdoor applications. Such materials include metals, protective coatings (paint, electroplating, and anodizing), plastics, and rubbers. Accelerating stresses include solar radiation (wavelength and intensity) and chemicals (humidity, salt, sulphur, and ozone). The degradation generally involves corrosion, oxidation (rust), tarnishing, or other chemical reaction.

8.2.1.2. Stresses and Stress Levels

Accelerated life test stresses and stress levels should be chosen so that they accelerate the failure modes under consideration but do not introduce failure modes that would never occur under use conditions. Normally, these stress levels will fall outside the product specification limits but inside the design limits as illustrated in the figure below:



This choice of stresses and stress levels and the process of setting up the experiment is of the utmost importance. The design engineer(s) and material scientist(s) are consulted to determine what stimuli (stress) is appropriate as well as to identify the appropriate limits (or stress levels). If these stresses or limits are unknown, multiple tests with small sample sizes can be performed in order to ascertain the appropriate stress(es) and stress levels. Proper use of Design of Experiments (DOE) methodology is also crucial at this step. In addition to proper stress selection, the application of the stresses must be accomplished in some logical, controlled and quantifiable fashion. Accurate data on the stresses applied as well as the observed behavior of the test specimens must be maintained. It is clear that as the stress used in an accelerated test becomes higher the required test duration decreases. However, as the stress level moves farther away from the use conditions, the uncertainty in the extrapolation increases. Confidence intervals provide a measure of the uncertainty in extrapolation.

8.2.1.3. Stress Loading

The stress loading in an accelerated test can be applied various ways. They include constant, cyclic, step, progressive, and random stress loading.

Constant stress. The most common stress loading is constant stress. Each specimen is run at a constant stress level. Figure depicts a constant stress test with three stress levels. There the history of a specimen is depicted as moving along a horizontal line until it fails at a time shown by an×. An unfailed specimen has its age shown by an arrow. At the highest level, all four specimens ran to failure. At the middle level, four ran to failure, and one was unfailed. At the lowest level, four ran to failure, and four were unfailed. In use, most products run at constant stress. Then a constant stress test mimics actual use. Moreover, such testing is simple and has advantages. First, in most tests, it is easier to maintain a constant stress level. Second, accelerated test models for constant stress are better developed and empirically verified for some materials and products. Third, data analyses for reliability estimation are well developed and computerized.

Step stress. In step-stress loading, a specimen is subjected to successively higher levels of stress. A specimen is first subjected to a specified constant stress for a specified length of time. If it does not fail, it is subjected to a higher stress level for a

specified time. The stress on a specimen is thus increased step until it fails. Usually all specimens go through the same specified pattern of stress levels and test times. Sometimes different patterns are applied to different specimens. Figure depicts two such patterns. Such data may be censored. Pattern 1 has six failures and three runouts.

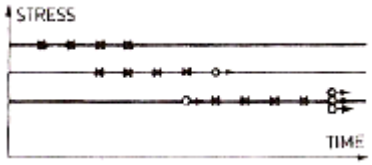


Figure 8.1: Constant stress test (× failure, O→ run out).

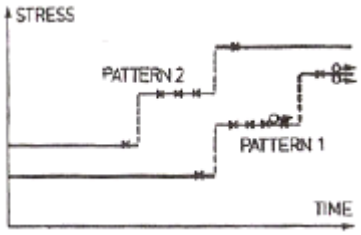


Figure 8.2: Step-stress test (× failure, O→ runout).

Advantages. The main advantage of a step-stress test is that is quickly yields failures. The increasing stress levels ensure this. Statisticians are happy to have failures, because they yield estimates of the model and of the product life. Engineers are happier when there are no failures, which suggest (perhaps incorrectly) that the product is reliable. Quick failures do not guarantee more accurate estimates. A constant stress test with a few specimen failures usually yields greater accuracy than a shorter step-stress test where all specimens fail. Roughly speaking, the total time on test (summed over all specimens) determines accuracy-not the number of failures.

Disadvantages. There is a major disadvantage of step-stress tests for reliability estimation. Most products run at constant stress-not step stress. Thus the model must properly take into account the cumulative effect of exposure at successive stresses. Moreover, the model must also provide an estimate of life under constant stress. Such a model is more complex than one for a constant stress test. Thus, constant stress tests are generally recommended over step-stress tests for reliability estimation. Another disadvantage of a step-stress test is that failure modes occurring at high stress levels (in later steps) may differ from those at use conditions.

Progressive stress. In progressive stress loading, a specimen undergoes a continuously increasing level of stress. Different groups of specimens may undergo

different progressive stress patterns. Figure 8.3 depicts such a test with three patterns-each a linearly increasing stress. As shown in Figure, under a low rate of rise of stress, specimens tend to live longer and to fail at lower stress. Such life data may be censored. In metal fatigue, such a test with a linearly increasing mechanical load is called a **Prot test.**

Disadvantages. Progressive stress tests have the same disadvantages as step-stress levels. Moreover, it may be difficult to control the progressive stress accurately enough. Thus constant stress tests are generally recommended over progressive stress tests for reliability estimation.

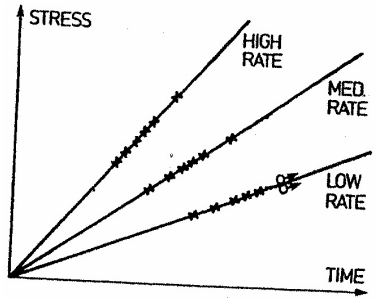


Figure 8.3: Progressive stress test (× failure, O→ runout).

Cyclic stress. In use, some products repeatedly undergo a cyclic stress loading. For example, insulation under ac voltage sees a sinusoidal stress. Also, for example, many metal components repeatedly undergo a mechanical stress cycle. A **cyclic stress** test for such a product repeatedly loads a specimen with the same stress pattern at high stress levels. Figure depicts a cyclic stress test. For many products, a cycle is sinusoidal. For others, the duty (or test) cycle repeats but is not sinusoidal. The (usually high) number of cycles to failure is the specimen life. Such life data may be censored.

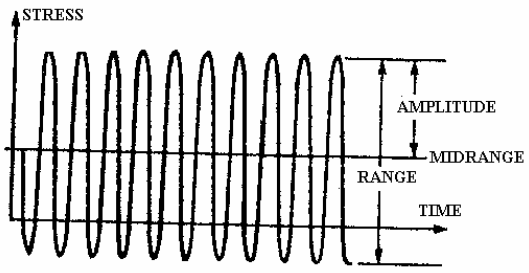


Figure 8.4: Cyclic-stress loading.

Examples

Insulation. For insulation tests, the stress level is the *amplitude* of the ac voltage sinusoid, which alternates from positive to negative voltage. So a single number characterizes the level. For purpose of modeling and data analysis, such cyclic stress is regarded as a constant, where the vertical axis shows the voltage amplitude.

Metals. In metal fatigue tests, usually a specimen undergoes (nearly) sinusoidal loading. But the sinusoid need not have a mean stress of zero. Figure 8.4 shows such a sinusoid with a positive mean. Tensile stress is positive, and compressive stress is negative in the figure. Thus, according to the figure, the specimen is under tension for most of a cycle and under compression for a small part of a cycle. Such sinusoidal loading is characterized by two numbers, say, the stress range and the mean stress. Frequency often has negligible effect. Thus, fatigue life can be regarded as a function of these two "constant" stress variables. In place of the stress range, metallurgists use the A-ratio; it is the stress amplitude (half the range) divided by the mean stress. For example, suppose a specimen is cycled from 0 psi to 80,000 psi compression and back to 0 psi. The mean stress is 40,000 psi, and the A-ratio is 0.5(80,000-0)/40,000 = 1. The A-ratio for ac voltage cycling of insulation is infinity, since the mean voltage is zero.

Random stress. Some products in use undergo randomly changing levels of stress, as depicted in Figure 8.5. For example, bridge members and air-plane structural components undergo wind buffeting. Also, environmental stress screening uses random vibration. Then an accelerated test typically employs random stresses with the same distribution as actual random stresses but at higher levels. Like cyclic stress tests, random stress models employ some characteristics of the stress distribution as stress variables (say, the mean, standard deviation, correlation function, and power spectral density). Then such a test is regarded as a constant stress test; this, of course, is simplistic but useful. The test can then be depicted as in Figure 8.6 where the horizontal line shows the mean stress. Moreover, specimen life is modeled with a constant stress model, and the data are analyzed accordingly. Such life data may be censored.

Figure 8.5: Random stress loading

8.3. Types of Accelerated Test Data

Accelerated test data can be divided into two types. Namely, the product characteristic of interest is 1) life or 2) some other measure of performance, such as tensile strength or ductility.

Performance data: One may be interested in how product performance degrades with age. In such performance testing, specimens are aged under high stress, and their performance measured at different ages. Such performance data are analyzed by fitting a degradation model to the data to estimate the relationship between performance, age, and stress.

Life data: The proper analysis of life data depends on the type of data. The following paragraphs describe the common types of life data from a single test or design condition.

Complete data: Complete data consist of the exact life (failure age) of each sample unit. Figure 8.6 A depicts a complete sample from a single test condition. There the length of a line corresponds to the length of life of a test unit. Much life data are incomplete. That is, the exact failure times of some units are unknown, and there is only partial information on their failure times. Examples are given below.

Censored: Often when life data are analyzed, some units are unfailed, and their failure times are known only to be beyond their present running times. Such data are said to be **censored on the right.** In the literature, such data or tests are called **truncated.** Unfailed units are called run-outs, survivors, removals, and suspensions. Such censored data arise when some units are (1) removed from test or service before they fail, (2) still running at the time of the data analysis, or (3) removed from test or service because they failed from an extraneous cause such as test equipment failure. Similarly, a failure time known only to be before a certain time is said to be **censored on the left.** If all unfailed units have common running time and all failure

times are earlier, the data are said to be **singly censored** on the right. Singly censored data arise when units are started together at a test condition and the data are analyzed before all units fail. Such data are singly **time censored** if the censoring time is fixed; then the number of failures in that fixed time is random. Figure 8.6 B depicts such a sample. There the line for an unfailed unit shows how long it ran without failure and the arrow pointing to the right indicates that the unit's failure time is later. Time censored data are also called **Type I censored.** Data are singly **failure censored** if the test is stopped when a specified number of failures occur. The time to that fixed number of failures is random. Figure 8.6 C depicts such a sample. Time censoring is more common in practice. Failure censoring is more common in the theoretical literature, as it is mathematically more tractable.

Multiply censored. Much data censored on the right have differing running times intermixed with the failure times. Such data are called **multiply censored** (also progressively, hyper-, and arbitrarily censored). Figure 8.6 D depicts such a sample. Multiply censored data arise when units go on test at different times. Thus they have different running times when the data are recorded. Such data may be time censored (running times differ from failure times, as shown in Figure 8.6 D) or failure censored (running times equal failure times, as shown in Figure 8.6 E).

Competing modes: A mix of competing failure modes occurs when sample units fail from different causes. Figure 8.6 F depicts such a sample, where A, B, and C denote different failure modes. Data on a particular failure mode consist of the failure times of units failing by that mode. Such data for a mode are multiply censored.

Quantal-response: Sometimes one knows only whether the failure time of a unit is before or after a certain time. Each observation is either censored on the right or else on the left. Such life data arise if each unit is inspected **once** to see if it has already filed or not. Such inspection data are called **quantal-response** data, also called sensitivity, probit, binary, and all-or-nothing response data. Figure 8.6 G depicts such a sample. There the arrow for each unit shows whether the unit failed before its inspection or will fail later.

Interval: When each unit is inspected for failure **more than once,** one knows only that a unit failed in an interval between inspections. So-called **interval,** grouped, or

read-out data are depicted in Figure 8.6 H. There a solid line shows the interval where a unit failed, and a dotted line shows an inspection interval where a unit failed, and a dotted line shows an inspection interval where it did not fail. Such data can also contain right and left censored observations.

Purpose: Analyses of such censored and intervals data have much the same purpose as analyses of complete data, for example, estimation of model parameters and the product life distribution and prediction of future observations.

To understand the process involved with extrapolating from overstress test data to use level conditions, let's look closely at a simple accelerated life test. For simplicity we will assume that the product was tested under a single stress at a single constant stress level. We will further assume that times-to-failure data have been obtained at this stress level. The times-to-failure at this stress level can then be easily analyzed using an underlying life distribution. A *pdf* of the times-to-failure of the product can be obtained at that single stress level using traditional approaches. This *pdf,* the overstress *pdf*, can likewise be used to make predictions and estimates of life measures of interest at that particular stress level. The objective in an accelerated life test, however, is not to obtain predictions and estimates at the particular elevated stress level at which the units were tested, but to obtain these measures at another stress level, the use stress level.

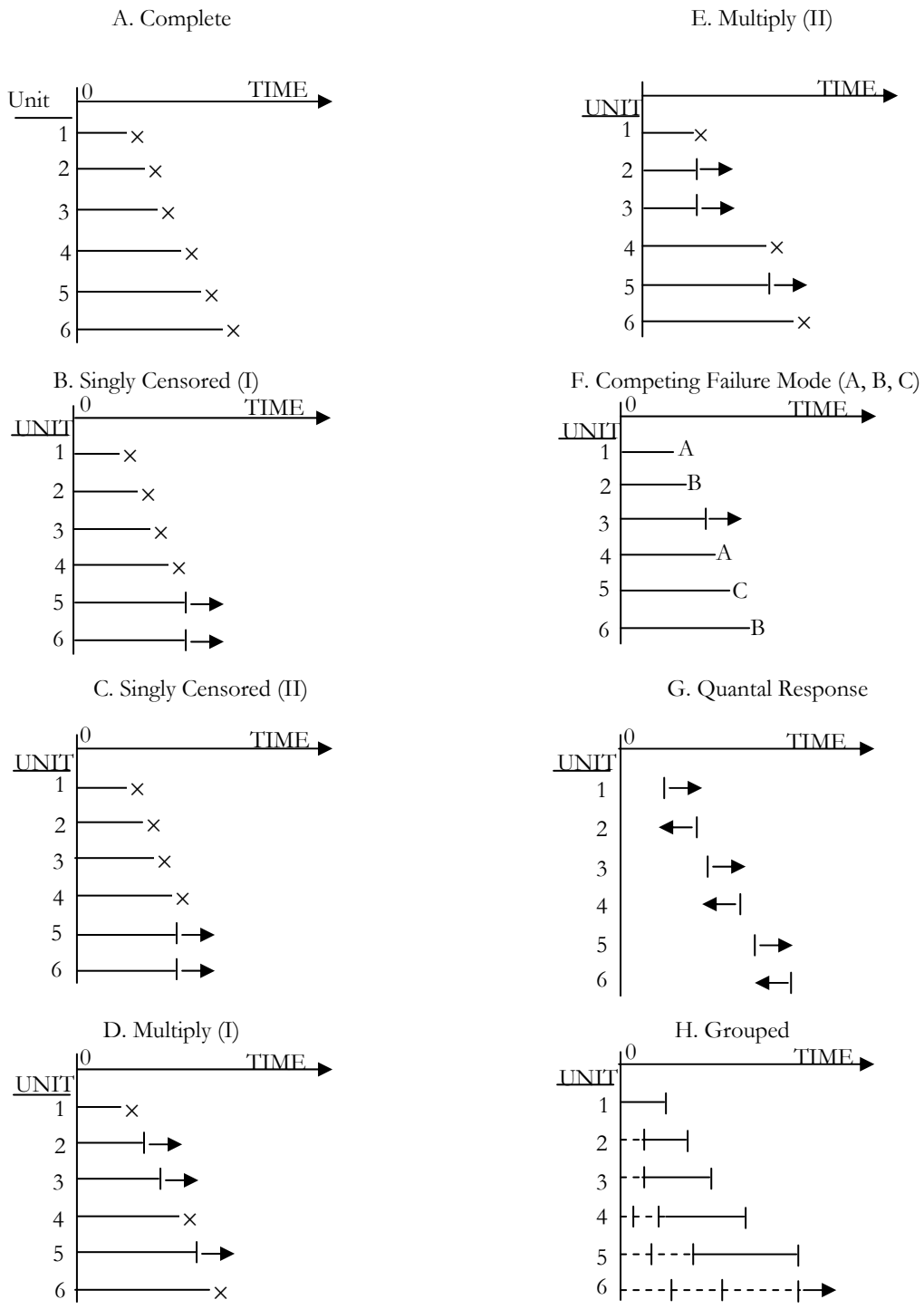
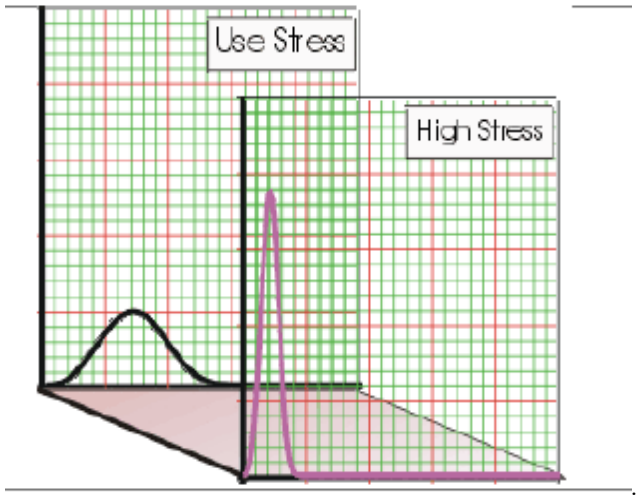


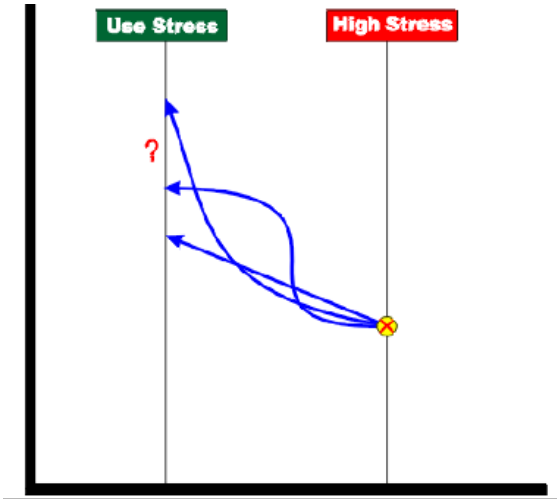
Figure 8.6: Types of Accelerated Data Types of data (failure time×, running time ⊢▶ , failure occurred earlier ◀⊣)

To accomplish this objective, we must devise a method to traverse the path from the overstress *pdf* to extrapolate a use level *pdf*.

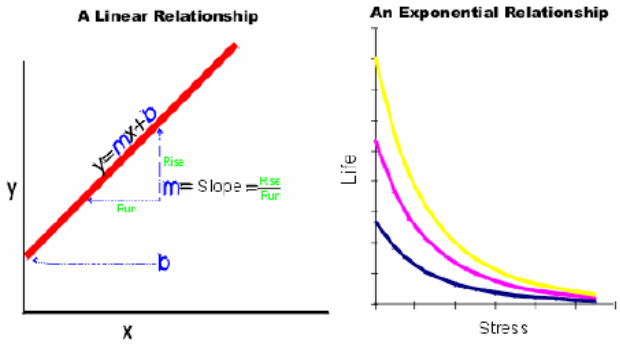
A typical behavior of the *pdf* at the high stress (or overstress level) and the *pdf* at the use stress level is shown below:



To further simplify the scenario, let's assume that the *pdf* for the product at any stress level can be described by a single point as shown. In the figure, we need to determine a way to project (or map) this single point from the high stress to the use stress.



Obviously, there are infinite ways to map a particular point from the high stress level to the use stress level. We will assume that there is some model (or a function) that maps our point from the high stress level to the use stress level. This model or function can be described mathematically and can be as simple as the equation for a line (A simple models or relationships).



Even when a model is assumed (*i.e.* linear, exponential, etc.), the mapping possibilities are still infinite since they depend on the parameters of the chosen model or relationship. For example, a simple linear model would generate different mappings for each slope value because we can draw an infinite number of lines through a point. If we tested specimens of our product at two different stress levels, we could begin to fit the model to the data. Obviously, the more points we have, the better off we are in correctly mapping this particular point or fitting the model to our data.

8.4. Analysis Method

With our current understanding of the principles behind accelerated life testing analysis, we will continue with a discussion of the steps involved in performing an analysis on life data that has been collected from accelerated life tests like Quantitative Accelerated Life.

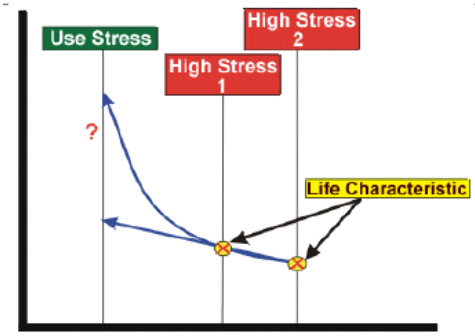
Select a Life Distribution

The first step in performing an accelerated life data analysis is to choose an appropriate life distribution. Although it is rarely appropriate, the exponential distribution, because of its simplicity, has in the past been widely used as the

underlying life distribution. The Weibull and lognormal distributions, which require more involved calculations, are more appropriate for most uses.

Select a Life-Stress Relationship

After you have selected an underlying life distribution appropriate to the data, the second step is to select (or create) a model that describes a characteristic point or a life characteristic of the distribution from one stress level to another.



The life characteristic can be any life measure such as the mean, median, $R(x)$, $F(x)$, etc. This life characteristic is expressed as a function of stress. Depending on the assumed underlying life distribution, different life characteristic are considered. Typical life characteristics for some distributions are shown in the table below.

Distribution	*Parameters*	*Life Characteristic*
Weibull	β^*, η	Scale parameter, η
Exponential	λ	Mean Life ($1/\lambda$)
Lognormal	$\overline{T}$, σ^*	Median, $\breve{T}$

**Usually assumed constant*

8.4.1. Life-Stress Models

There are three types of models for relating the failure data at accelerated conditions to reliability measures at normal (or design) stress conditions. The underlying assumption in relating the failure data when using any of the models is that the components (or products) operating under the normal conditions experience the same failure mechanism as those occurring at the accelerated stress conditions. For example, if the macroscopic examination of the fracture surface of the failed components indicates that fatigue cracking initiated at a corrosion pit is the cause of the failure at normal operating conditions, then the accelerated test should be designed so that the failure mechanism is identical to that of the normal conditions.

Models can be classified as statistics-based models (parametric and nonparametric), physics-statistics-based models, and physics-experimental-based

models. In all of these models, we assume that the stress levels applied at the accelerated conditions are within a range of true acceleration-that is, if the failure-time distribution at a high stress level is known and time-scale transformation to the normal conditions is also known, we can mathematically derive the failure-time distributions at normal operating conditions (or any other stress level). For practical purposes, we assume that the time-scale transformation (also referred to the acceleration factor, $A_F > 1)$ is constant, which implies that we have a true linear acceleration. Let the subscripts 0 and s refer to the operating conditions and stress conditions, respectively. Thus,

- The relationship between the time to failure at operating conditions and stress conditions is

$$t_o = A_F \times t_s. \tag{8.1}$$

- The cumulative distribution functions are related as

$$F_o(t) = F_s\left(\frac{t}{A_F}\right). \tag{8.2}$$

- The probability density functions are related as

$$f_o(t) = \left(\frac{1}{A_F}\right) f_s\left(\frac{t}{A_F}\right). \tag{8.3}$$

- The failure rates are given by

$$h_o(t) = \frac{f_o(t)}{1 - F_o(t)}.$$

$$= \frac{\left(\frac{1}{A_F}\right) f_s\left(\frac{t}{A_F}\right)}{1 - F_s\left(\frac{t}{A_F}\right)}$$

$$h_o(t) = \left(\frac{1}{A_F}\right) h_s\left(\frac{t}{A_F}\right). \tag{8.4}$$

Statistics-based models are generally used when the exact relationship between the applied stresses (temperature, humidity, voltage) and the failure time of the component (or product) is difficult to determine based on physics or chemistry principles. In this case, components are tested at different accelerated stress levels $s_1, s_2, \ldots$. The failure times at each stress level are then used to determine the most

appropriate failure-time probability distribution along with its parameters. As stated earlier, the failure times at different stress levels are linearly related to each other. Moreover, the failure times stress level s_1 is expected to be the same at different stress levels $s_2, s_3, \ldots$ as well as at the normal operating conditions. The shape parameters of the distributions are the same for all stress levels (including normal conditions), but the scale parameters may be different.

8.4.2. Statistics Based Models

8.4.2.1. Exponential Distribution Acceleration Model

This is the case where the time to failure at an accelerated stress s is exponentially distributed with parameter λ_s. The hazard rate at the stress is constant. The CDF at stress s is:

$$F_s(t) = 1 - e^{-\lambda_s t} \tag{8.5}$$

Following Eq. (8.2), the CDF at the normal operating conditions is

$$F_o(t) = F_s\left(\frac{t}{A_F}\right) = 1 - e^{\frac{-\lambda_s t}{A_F}} . \tag{8.6}$$

Similarly,

$$\lambda_o = \frac{\lambda_s}{A_F} \tag{8.7}$$

The failure rate at stress level s can be estimate for both non-censored and censored failure data as follows:

$$\lambda_s = \frac{n}{\sum_{i=1}^{n} t_1} \text{ for non-censored data ; } \textit{and}$$

$$\lambda_s = \frac{r}{\sum_{i=1}^{r} t_1 + \sum_{i=1}^{n-r} t_i^+} \text{ for censored data,}$$

where t_i is the time of the ith failure, t_i^+ is the ith censoring time, n is the total number of units under test as stress s, and r is the number of failed units at the accelerated stress s.

Typical accelerated-testing plans allocate equal units to the test stresses. However, units tested at stress levels close to the design or operating conditions may

not experience enough failures that the can be effectively used in the acceleration models. Therefore, it is preferred to allocate more test units to the low-stress conditions than to the high-stress conditions so as to obtain an equal expected number of failures at each condition.

8.4.2.2. Weibull Distribution Acceleration Model

Consider the true linear acceleration case. The relationships between the failure-time distributions at the accelerated and normal conditions can be derived using Eqs. (8.2) and (8.3). Thus

$$F_s(t) = 1 - e^{-\left(\frac{t}{\theta_s}\right)^{\gamma_s}} \quad t \geq 0, \gamma_s \geq 1,\ \theta_s > 0$$

and

$$F_o(t) = F_s\left(\frac{t}{A_F}\right) = 1 - e^{\left(\frac{t}{A_F\theta_s}\right)^{\gamma_s}} = 1 - e^{-\left(\frac{t}{\theta_o}\right)^{\gamma_o}}. \tag{8.8}$$

The underlying failure-time distributions at both the accelerated stress and operating conditions have the same shape parameters-that is, $\gamma_s = \gamma_o$, and $\theta_o = A_F\theta_s$. If the shape parameters at different stress levels are significantly different, then either the assumption of true linear acceleration is invalid or the Weibull distribution is inappropriate to use for analysis of such data.

Let $\gamma_s = \gamma_o = \gamma \geq 1$ >. Then the probability density function at normal operating conditions is

$$f_o(t) = \frac{\gamma}{A_F\theta_s}\left(\frac{t}{A_F\theta_s}\right)^{\gamma-1} \exp -\left(\frac{t}{A_F\theta_s}\right)^{\gamma} \quad t \geq 0, \theta_s \geq 0. \tag{8.9}$$

The MTTF at normal operating conditions is

$$MTTF_o = \theta_o\Gamma\left(1 + \frac{1}{\gamma}\right). \tag{8.10}$$

The hazard rate at the normal conditions is

$$h_o(t) = \frac{\gamma}{A_F\theta_s}\left(\frac{t}{A_F\theta_s}\right)^{\gamma-1} = \frac{h_s(t)}{A_F{}^{\gamma}}. \tag{8.11}$$

8.4.3. Physics Statistics Based Models

The physics-statistics-based models utilize the effect of the applied stresses on the failure rate of the units under test. For example, the failure rate of many integrated circuits is accelerated by temperature and the model that relates the failure rate with temperature should reflect the physical and chemical properties of the units. Moreover, since several units are usually tested at the same stress level and all times of failure are random events, the failure-rate expression should also reflect the underlying failure-time distribution. Thus, physics-statistics-based models are needed to describe the failure-rate relationships.

8.4.3.1. The Arrhenius Model

Elevated temperature is the most commonly used environmental stress for accelerated life testing of microelectronic devices. The effect of temperature on the device is generally modeled using the Arrhenius reaction rate equation given by

$$r = A_e^{-(E_a/kT)}, \tag{8.12}$$

where,

r = the speed of reaction,
A = an unknown nonthermal constant,
E_a = the activation enery (eV); energy that a molecule must have before it can taken part in the reaction,
k = the Boltzmann Constant $(8.623\times10^{-5} eV/K)$, and
T = the temperature in Kelvin.

Activation energy (E_a) is a factor that determines the slope of the reaction rate curve with temperature-that is, it describes the acceleration effect that temperature has on the rate of a reaction and is expressed in electronic volts (eV). For most applications, E_a is treated as a slope of a curve rather than a specific energy level. A low value of E_a indicates a small slope or a reaction that has a small dependence on temperature. On the other hand, a large value of E_a indicates a high degree of temperature dependence.

Assuming that device life is proportional to the inverse reaction rate of the process, then Eq. (8.12) can be rewritten as

$$L = Ae^{+(E_a/kT)}.$$

The lives of the units at normal operating temperature L_o and accelerated temperature L_s are related by:

$$\frac{L_o}{L_s} = \frac{e^{(E_a/kT_o)}}{e^{(E_a/kT_s)}}$$

or

$$L_o = L_s \exp\frac{E_a}{k}\left(\frac{1}{T_o} - \frac{1}{T_s}\right). \tag{8.13}$$

When the mean life L_o at normal operating conditions is calculated and the underlying life distribution is exponential, then the failure rate a normal operating temperature is

$$\lambda_o = \frac{1}{L_o},$$

and the thermal acceleration factor is

$$A_T = \frac{L_o}{L_s}$$

or

$$A_T = \exp\left[\frac{E_a}{k}\left(\frac{1}{T_o} - \frac{1}{T_s}\right)\right]. \tag{8.14}$$

8.4.3.2. The Eyring Model

The Eyring model is similar to the Arrhenius model. Therefore, it is commonly used for modeling failure data when the accelerated stress is temperature. It is more general than the Arrhenius model since it can model data from temperature acceleration testing as well as data from other single stress testing such as electric field. The Eyring model for temperature acceleration is

$$L = \frac{1}{T}\exp\left[\frac{\beta}{T} - \alpha\right], \tag{8.15}$$

Where, α and β are constants determined from the accelerated test data, L is the mean life, and T is the temperature in Kelvin. As shown in Eq. (8.15), the underlying failure time distribution is exponential. Thus the hazard rate λ and $1/L$. The

relationship between lives at the accelerated conditions and the normal operating conditions is obtained as follows. The mean life at accelerated stress conditions is

$$L_s = \frac{1}{T_s}\exp\left[\frac{\beta}{T_s} - \alpha\right]. \tag{8.16}$$

The mean life at normal operating conditions is

$$L_o = \frac{1}{T_o}\exp\left[\frac{\beta}{T_o} - \alpha\right]. \tag{8.17}$$

Dividing Eq. (8.16) by Eq. (8.17), we obtain

$$L_o = L_s\left(\frac{T_s}{T_o}\right)\exp\left[\beta\left(\frac{1}{T_o} - \frac{1}{T_s}\right)\right]. \tag{8.18}$$

The acceleration factor is

$$A_F = \frac{L_o}{L_s}.$$

Equation (8.18) is identical to the result of the Arrhenius model given in Eq. (8.13) with the exception that the ratio $\left(T_s / T_o\right)$ of the nonexponential curve in Eq. (8.18) is set to equal 1. In this case, β reduces to be the ratio between E_a and k (Boltzmann's constant).

The constant α and β can be obtained through the maximum likelihood method, by solving the following two equations for l samples tested at different stress levels and r_i failures $\left(i = 1, 2, \ldots, l\right)$ are observed at stress level V_i. The equations are the resultants of taking the derivatives of the likelihood function with respect to α and β, respectively and equating them to zero.

$$\sum_{i=1}^{l} R_i - \sum_{i=1}^{l}\left[R_i / \left(\hat{\lambda}_i V_i\right)\right]\exp\left[\alpha - \beta\left(V_i^{-1} - \bar{V}\right)\right] = 0 \tag{8.19}$$

$$\sum_{i=1}^{l}\left(R_i / \left(\hat{\lambda}_i V_i\right)\right)\left(V_i^{-1} - \bar{V}\right)\exp\left[\alpha - \beta\left(V_i^{-1} - \bar{V}\right)\right] = 0, \tag{8.20}$$

Where,

$\hat{\lambda}_i$ = the estimated hazard rate at stress V_i,

$$R_i = \begin{cases} r_i & \text{if the location of the parameter is known,} \\ r_i - 1 & \text{if the location of the parameter is unknown,} \end{cases}$$

$$\bar{V} = \frac{\sum_{i=1}^{l} \frac{R_i}{V_i}}{\sum_{i=1}^{l} R_i}$$

V = stress variable. If temperature, then V is in Kelvin.

8.4.3.3. The Inverse Power Rule Model

The energy power rule model is derived based on the Kinetic theory and activation energy. The underlying life distribution of this model is Weibull. The mean time to failure (life) decreases as the nth power of the applied stress (usually voltage). The inverse power law is expressed as:

$$L_s = \frac{C}{V_s^n} \quad C > 0, \tag{8.21}$$

Where, L_s is the mean life at the accelerated stress V_s and C and n are constants. The mean life at normal operating conditions is

$$L_o = \frac{C}{V_o^{n\cdot}} \tag{8.22}$$

Thus,

$$L_o = L\left(\frac{V_s}{V_{\hat{o}}}\right)^n. \tag{8.23}$$

$L_o = L\left(\frac{V_s}{V_{\hat{o}}}\right)^n$. amended Eq.(8.21) without changing its basic characteristic to

$$L_i = \frac{C}{\left(V_i / \bar{V}\right)^n}, \tag{8.24}$$

where L_i is the mean life at stress level V_i and $\bar{V}$ is the weighted geometric mean of the V_i's and is expressed as

$$\bar{V} = \prod_{i=1}^{k} (V_i)^{R_i / \sum_{i=1}^{k} R_i}, \tag{8.25}$$

where $R_i = \gamma_i$ (number of failures at stress V_i) or $R_i = \gamma_i - 1$ depending on whether or not the shape parameter of the failure time distribution is known. The likelihood function of C and n is

$$\prod_{i=1}^{k} \Gamma^{-1}(R_i) \left[\frac{R_i}{C} \left(\frac{V_i}{\bar{V}} \right)^n \right]^{R_i} \left(\hat{L}_i \right)^{R_i - 1} \exp\left[-\frac{R_i \hat{L}_i}{C} \left(\frac{V_i}{\bar{V}} \right)^n \right],$$

where $\hat{L}_i$ is the estimated mean life at stress V_i. The maximum likelihood estimators of $\hat{C}$ and $\hat{n}$ are obtained by solving the following two equations:

$$\hat{C} = \frac{\sum_{i=1}^{k} R_i \hat{L}_i \left(V_i / \bar{V} \right)^{\hat{n}}}{\sum_{i=1}^{k} R_i} \tag{8.26}$$

$$\sum_{i=1}^{k} R_i \hat{L}_i \left(\frac{V_i}{\bar{V}} \right)^{\hat{n}} \log \frac{V_i}{\bar{V}} = 0. \tag{8.27}$$

The asymptotic variance of $\hat{n}$ and $\hat{C}$

$$\sigma_n^2 = \left[\sum_{i=1}^{k} R_i \left(\log \frac{V_i}{\bar{V}} \right)^2 \right]^{-1} \tag{8.28}$$

$$\sigma_c^2 = C^2 \left(\sum_{i=1}^{k} R_i \right)^{-1}. \tag{8.29}$$

8.4.3.4. Combination Model

This model is similar to the Eyring multiple stress model when temperature and another stress such as voltage are used in the accelerated life test. The essence of the model is that the Arrhenius reaction model and the inverse power rule model are combined to form this combination model. It is valid when the shape parameter of the Weibull distribution is equal to one in the inverse power rule model. The model is given by

$$\frac{L_o}{L_s} = \left(\frac{V_o}{V_s}\right)^{-n} \exp\left[E_a / k\left(\frac{1}{T_o} - \frac{1}{T_s}\right)\right], \tag{8.30}$$

Where:

L_o = the life at normal operating conditions,
L_s = the life at accelerated stress conditions,
V_o = the normal operating volt,
V_s = the accelerating stress volt,
T_s = the accelerated stress temperature, and
T_o = the normal operating temperature.

8.4.4. Physics Experimental Based Models

The time to failure of many devices and components can be estimated based on the physics of the failure mechanism by either the development of theoretical basis for the failure mechanisms or the conduct of experiments using different levels of the parameters that affect the time to failure. There are many failure mechanisms resulting from the application of different stresses at different levels. For example, the time of failure (TTF) of packaged silicon integrated circuits due to the electromigration phenomenon is affected by the current density through the circuit and by the temperature of the circuit. Similarly, the time to failure of some components may be affected by relative humidity only.

The following sections present the most widely used models for predicting the time to failure as a function of the parameters that result in device or component failures.

8.4.4.1. Electromigration Model

Electromigration is the transport of microcircuit current conductor metal atoms due to electron wind effects. If, in an aluminum conductor, the electron current density is sufficiently high, an electron wind effect is created. Since the size and mass of an electron are small compared to the atom, the momentum imparted to an aluminum atom by an electron collision is small . If enough electrons collide with an aluminum atom, then the aluminum atom will move gradually causing a depletion at the negative end of the conductor. This will result in voids or hillocks along the conductor, depending on the local microstructure, causing a catastrophic failure. The median time to failure (MTF) in the presence of electromigration is given equation:

$$MTF = AJ^{-n}e^{E_a/kT,} \tag{8.31}$$

where A, n are constants, J is the current density, k is Boltzmann's constant, T is the absolute temperature, and E_a is the activation energy (0.6 eV for aluminum and 0.9 eV for gold). The electromigration exponent n ranges from 1 to 6.

In order to determine the lives of components at normal operating conditions, we perform accelerated life testing on samples of these components by subjecting them to different stresses. In the case of electromigration, the stresses are the electric current and the temperature. From three or more stress conditions, the electromigration parameters such as E_a and n can be obtained.

For a fixed current, we can estimate the median life at the operating temperature as

$$\frac{t_{50}(T_o)}{t_{50}(T_s)} = \exp\left[\frac{E_a}{k}\left(\frac{1}{T_o} - \frac{1}{T_s}\right)\right], \tag{8.32}$$

where $t_{50}(T_i)$ is the median life at $T_i (i = o \text{ or } s)$.

Similarly, we can fix the temperature and vary the current density. Thus,

$$\frac{t_{50}(J_o)}{t_{50}(J_s)} = \left(\frac{J_o}{J_s}\right)^{-n}$$

8.4.4.2. Humidity Dependence Failures

Corrosion in a plastic integrated circuit may deteriorate the leads outside out side the encapsulated circuit or the metallization interconnect inside the circuit. The basic ingredients needed for corrosion are moisture (humidity) and ions for the formation of an electrolyte, and metal for electrodes and an electric field. If any of these is missing, corrosion will not take place.

The general humidity model is

$$t_{50} = A(RH)^{-\beta} \quad \text{or} \quad t_{50} = Ae^{-\beta(RH)},$$

Where, t_{50} is the median life of the device, A and β are constants, and RH is the relative humidity. However, conducting an accelerated test for only humidity requires years before meaningful results are obtained. Therefore, temperature and humidity

are usually combined for life testing, which is referred to as highly accelerated stress testing (HAST). The most common form of HAST is the 85/85 test where devices are tested at a relative humidity of 85 percent and a temperature of 85°C. Voltage stress is usually added to this stress in order to reduce the duration of the test further. The time to failure of a device operating under temperature, relative humidity, and voltage conditions is expressed as

$$t = ve^{\frac{E_a}{kT}} e^{\frac{\beta}{RH}}, \tag{8.33}$$

Where:

$t =$ the time to failure,
$v =$ the applied voltage,
$E_a =$ the activation energy,
$k =$ Boltzmann's constant,
$T =$ the absolute temperature,
$\beta =$ a constant, and
$RH =$ the relative humidity.

Let the subscripts s and o represent the accelerated stress conditions and the normal operating conditions, respectively. The acceleration factor is obtained as

$$A_F = \frac{t_o}{t_s} = \frac{v_o}{v_s} e^{\frac{E_a}{k}\left[\frac{1}{T_o} - \frac{1}{T_s}\right]} e^{-\beta\left[\frac{1}{RH_o} - \frac{1}{RH_s}\right]}. \tag{8.34}$$

Changes in the microelectronics require that the manufacturers consider faster methodologies to detect failures causes by corrosion. Some manufacturers use pressure cookers to induce corrosion failures in a few days of days of test time. Studies showed that pressurized humidity test environments forced moisture into the plastic encapsulant much more rapidly than other types of humidity test methods.

8.4.4.3. Temperature-Humidity Relationship

When performing accelerated life testing analysis, a life distribution and a life-stress relationship are required. The temperature-humidity (T-H) relationship, a variation of the Eyring relationship, has been proposed for predicting the life at use conditions when temperature and humidity are the accelerated stresses in a test. This combination model is given by:

$$L(V,U) = Ae^{\frac{\phi}{V}+\frac{b}{U}} \tag{8.35}$$

where:

- ϕ is one of the three parameters to be determined,
- b is the second of the three parameters to be determined (also known as the activation energy for humidity),
- A is a constant and the third of the three parameters to be determined,
- U is the relative humidity (decimal or percentage),
- V is temperature (in absolute units).

The T-H relationship can be linearised and plotted on a life vs. stress plot. The relationship is linearised by taking the natural logarithm of both sides in Eqn. (8.35):

$$ln(L(V,U)) = ln(A) + \frac{\phi}{V} + \frac{b}{U} \tag{8.36}$$

Since life is now a function of two stresses, a life vs. stress plot can only be obtained by keeping one of the two stresses constant and varying the other one. Doing so will yield a straight line as described by Eqn. (8.36), where the term for the stress, which is kept at a fixed value, becomes another constant (in addition to the ln(A) constant).

8.4.4.4. Fatigue Failures

When repetitive cycles of stresses are applied to material, fatigue failures usually occur at a much lower stress than the ultimate strength of the material due to the accumulation of damage. Fatigue loading causes the material to experience cycles of tension and compressions, which result in crack initiations at the points of discontinuity, defects in material, or notches or scratches where stress concentration is high. The crack length grows as the repetitive cycles of stresses continue until the stress on the remaining cross-section area exceeds the ultimate strength of the material. At this moment, sudden facture occurs, causing instantaneous failure of the component or member carrying the applied stresses. It is important to recognize that the applied stresses are not only caused by applying physical load or force but also by temperature or voltage cycling. For example, *creep fatigue,* or the thermal expansion strains caused by thermal cycling, is the dominate failure mechanism causing breaks in surface mount technology (SMT)-solder attachments of printed circuits. Each

thermal cycle produces a specific net strain energy density in the solder that corresponds to a certain amount of fatigue damage. The long-term reliability depends on the cyclically accumulated fatigue damage in the solder, which eventually results in fracture . The reliability of components or devices subject to fatigue failure is often expressed in number of stress cycles corresponding to a given cumulative failure probability. A typical model for fatigue failure of a solder attachment is given by ,

$$N_f\left(x\%\right)=\frac{1}{2}\left[\frac{2\varepsilon}{F}\frac{h}{L_D\Delta\alpha\,\Delta T_e}\right]^{\frac{-1}{c}}\left[\frac{\ln\left(1-0.01x\right)}{\ln\left(0.5\right)}\right]^{\frac{1}{\beta}}, \tag{8.37}$$

Where:

$N_f\left(x\%\right)=$ number of cycles (fatigue life) that correspond to x percent failures,

$\varepsilon=$ the solder ductility,

$F=$ an experimental factor (Engelmaier, 1993),

h *and* $L_D=$ dimensions of the solder attachment,

$\Delta\alpha=$ a factor of the differences in the thermal expansion coefficient of component and substrate (that produces the stress),

$\Delta T_e=$ the effective thermal cycling range,

$c=$ a constant that relates the average temperature of the solder joint and the time for stress relaxation and creep per cycle, and

$\beta=$ 4 for the leadless surface mounted attachment.

8.4.5. Degradation Models

Most reliability data obtained from accelerated life testing are time-to-failure measurements obtained from testing samples of units at different stresses. However, there are many situations where the actual failure of the units, especially at stress levels close to the normal operating condition, may not fail catastrophically but degrade within the allotted test time. For example, a component may start a test with an acceptable resistance value reading, but during the test the resistance reading "drifts". As the test time progresses the resistance eventually reaches an unacceptable level that causes the unit to fail. In such cases, measurements of the degradation of the characteristics of interest are frequently taken during the test. The degradation data are then analyzed and used to predict the time of failure at normal conditions. It is obvious that there is no general degradation model that can be used for all devices or parameters for a specific device. For example, the degradation in the resistance of a device requires a model different from the one that measures degradation in the output current of the same device.

8.4.5.1. Resistor Degradation Model

The thin film integrated circuit resistor degradation mechanism can be described by:

$$\frac{\Delta R(t)}{R_0} = \left(\frac{t}{\tau}\right)^m, \tag{8.38}$$

Where,

$\Delta R(t)$ = the change in resistance at time t,
R_0 = the initial resistance,
t = time,
τ = the time required to cause 100 percent change in resistance, and
m = a constant.

The temperature dependence is embedded in τ as

$$\tau = \tau_0 e^{\frac{E_a}{kT}}, \tag{8.39}$$

Where, τ_0 is a constant.

Substituting Eq. (8.39) into Eq. (8.38) and taking the logarithm, we obtain

$$\ln\left(\frac{\Delta R(t)}{R_0}\right) = m\left[\ln(t) - \ln(\tau_0)\frac{E_a}{kT}\right]$$

or

$$\ln(t) = \ln(\tau_0) + \frac{1}{m}\ln\left(\frac{\Delta R(t)}{R_0}\right) + \frac{E_a}{kT}. \tag{8.40}$$

Once the constant m and τ_0 are determined we can use Eq. (8.38) to calculate the change in resistance at any time. The above equation can also be used to predict the life of a device subject to electromigration failures. Recall that the time to failure due to electromigration is given by Eq. (8.31). Taking the natural logarithm of Eq. (8.31) results in

$$\ln(MTF) = \ln(A) - n\ln(J) + \frac{E_a}{kT}. \tag{8.41}$$

Note that Eqs. (8.40) and (8.41) are identical.

The constant m and τ_0 can be obtained using the standard multiple regression.

Table 8.2 summarizes some of most frequently used models, their relevant parameters and applications.

Table 8.2: Frequently Used Acceleration Models, Their Parameters and Applications

Model	Description/Parameters	Model Equation	Application Examples
Arrhenius Acceleration Model	Life as function of temperature or chemical change	$L = A_0 \exp(\frac{E_a}{KT})$, where, L = life A_0 = scale factor determined by experiment e = base of natural algorithm Ea = Activation Energy (Unique for each mechanism) K = Boltzmann's constant = 8.62 X 10^{-5} eV/K T = Temperature in ^{0}K	Electrical Insulations and dielectrics, solid stste and semiconductors, intermettalic diffusion, Battery, lubricants 7 grease, plastics, Incandescent Lamp filaments
Inverse Power Law	Life as function of any given stress	Life at normal stress/Life at accelerated stress = (accelerated stress/normal stress)^N Where, N = Acceleration Factor	Electrical Insulation and Dielectrics (Voltage Endurance), Ball & Roller Bearings, Incandescent Lamp Filament, Flash lamps
Miner's Rule	Cumulative Linear fatigue Damage as a function of Flexing	CD = $\sum_{i=1}^{k} \frac{C_{Si}}{N_i} \leq 1$, where CD = Cumulative Damage C_{si} = Number of Cycles applied @ stress Si Ni = Number of cycles to failure under stress Si (Determined from S-N Diagram for that specific material) k = number of loads applied	Metal fatigue (valid only up to the yield strength of the material.
Coffin-Mansion	Fatigue Life of metals (Ductile materials) due to thermal cycling and/or thermal shock	Life = $\frac{A}{(\Delta T)^B}$, where Life = cycle to fail A, B = scale factor determined by experiment ΔT = Temperature Change	Solder joints and other connectors.
Peck's	Life as a combined function of temperature and humidity	$\tau = A_0 (RH)^{-2.7} \exp[\frac{0.79}{kT}]$, where τ = Median life (Time-to-failure) A0 = Scale factor determined by experiment. RH = Relative humidity.	Epoxy Packaging

Table 8.2: Frequently Used Acceleration Models, Their Parameters and Applications

Model	Description/Parameters	Model Equation	Application Examples
Peck's Power Law	Time-to Failure as a function of relative humidity, voltage and temperature	$TF = A_0 * RH^{-N} * f(V) * \exp(\frac{E_a}{kT})$, where TF = time to failure A0 = scale factor determined by experiment RH = Relative Humidity N = ~2.7 Ea = -0.7-0.8 eV(appropriate for Aluminum Corrosion when chlorides are present. f(V) = an unknown function of applied voltage.	Corrosion
Eyring/Black/Kenny	Life as a function of Temperature and Voltage (or Current density)	$\tau = \frac{A}{T}\exp(\frac{B}{kT})$, where τ = Median life (Time-to-failure) A, B = scale factor determined by experiment.	Capacitors, Electro migration in aluminum Conductors
Eyring	Time to failure as a function of current, electric fields and temperature	$TF = B(I_{sub})^{-N} \exp(\frac{E_a}{KT})$, where TF = time to failure B = scale factor determined by experiment. Isub = peak substrate current during stressing N = 2 to 4 Ea = -0.1 eV to –0.2 eV (Note that apparent activation energy is negative)	Hot carrier junction, surface inversion, mechanical stress
Therm0- Mechanical Stress	Time to failure as a function of change in temperature	$TF = B_0(T_0 - T)^{-n} \exp(\frac{E_a}{T})$, where B0 = Scale factor determined by the experiment. T0 = stress free temperature for metal (approximate metal deposition temperature for aluminum) n = 2-3 Ea = 0.5-0.6eV for grain boundary diffusion, ~1 eV for intra-grain diffusion	Stress generated by differing thermal expansion rates.

9. Repairable System Analysis

9.1. Availability and Maintainability Measures

A repairable system (RS) is a system, which after failure, can be restored to a functioning condition by some maintenance action other than replacement of the entire system. Note that replacing the entire system may be an option, but it is not the only option. Maintenance actions performed on a RS can be categorized in two ways. First, a maintenance action may be corrective (CM) or preventive (PM). CM actions are performed in response to system failures, whereas PM actions are intended to delay or prevent system failures. Note that PM actions may or may not be cheaper and/or faster than CM actions. Second, a maintenance action (CM or PM) may be a repair or a replacement. In our discussions, we assume that a RS is always in one of two states: functioning (up) or down. Note that a system may be down for CM or down for PM. The performance of a RS can be measured in several ways. We consider three categories of RS performance measures: (1) number of failures, (2) availability measures, (3) cost measures.

Availability and Reliability represent important performance parameters of a system, with respect to its ability to fulfill the required mission during a given functioning period. From this point of view, two main types of systems can be defined:

- Systems which must satisfy a specified mission within an assigned period of time: in this case reliability is the appropriate performance indicator of their ability to achieve the desired objective without failures;
- Systems maintained: in this case availability quantifies in a suitable way the system ability to fulfill the assigned mission at any specific moment of its life time. Basic maintenance procedures can be distinguished in: a. Off-schedule (corrective): this amounts to the replacement or repair of failed units; b. Preventive: this amounts to performing regular inspections, and possibly repair, following a given maintenance plan; c. Conditioned: it amounts to performing a repair action upon detection of degradation.

9.1.1. Contributions to unavailability

The main contributions to the unavailability of a system generally come from:

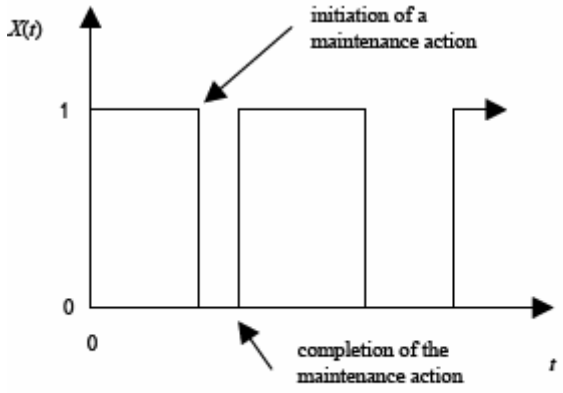
1. Unrevealed failure, i.e. when a stand-by component fails unnoticed. The system goes on without noticing the component failure until a test on the component is made or the component is demanded to function.
2. Testing/preventive maintenance, i.e. when a component is removed from the system because it has to be tested or must undergo preventive maintenance.
3. Repair, i.e. when a component is unavailable because under repair.

Let $N(t)$ denote the number of RS failures in the first t time units of system operation. Because of the stochastic (random) nature of RS behavior, $N(t)$ is random variable. Thus, we may focus our attention on the expected value, variance and probability distribution of $N(t)$.

9.2. Availability

Availability can be loosely defined as the proportion of time that a RS is in a functioning condition. However, there are four specific measures of availability found in the RS literature. All these measures are based on the RS status function:

$$X(t)=\begin{cases}1 & \text{if system is functioning at time t}\\ 0 & \text{if system is down at time } t\end{cases}$$



Graphical Portrayal of $X(t)$

The first of these availability measures is *Instantaneous availability or point availability*, $A(t)$ defined as the probability that a system (or component) will be operational (up and running) at any random time, t.

$$A(t) = \Pr[X(t) = 1]$$

The item functioned properly from 0 to t with probability $R(t)$ or it functioned properly since the last repair at time u, $0 < u < t$, with probability:

$$\int_0^t R(t-u)m(u)du$$

Then the point availability is the summation of these two probabilities, or:

$$A(t) = R(t) + \int_0^t R(t-u)m(u)du$$

With $m(u)$ being the renewal density function of the system, i.e., rate of change of the expected number of failures with respect to time.

By far the most commonly used availability measure, limiting availability is often easy to obtain. However, there are some cases in which limiting availability does not exist. The third availability measure is *average availability*, $A_{avg}(T)$ is the proportion of time during a mission or time period that the system is available for use. It represents the mean value of the instantaneous availability function over the period (0, T] and is given by:

$$A_{avg}(T) = \frac{1}{T}\int_0^T A(t)\,dt$$

Average availability corresponds to the average proportion of "uptime" over the first T time units of system operation. Since it is based on $A(t)$, average availability is typically difficult to obtain and rarely used in practice. However, because it captures availability behavior over a finite period of time, it is a valuable measure of RS performance. The other availability measure is *limiting average availability*, A_{avg}.

$$A_{avg} = \lim_{t\to\infty} A_{avg}(t)$$

When it exists, limiting average availability is almost always equivalent to limiting availability. To our knowledge, limiting average availability is never used in practice.

*Steady State Availability, t*he steady state availability of the system is the limit of the instantaneous availability function as time approaches infinity or:

$$A(\infty) = \lim_{t \to \infty} A(t)$$

(Note: For practical considerations, the instantaneous availability function will start approaching the steady state availability value after a time period of approximately four times the average time-to-failure.)

Inherent Availability is the steady state availability when considering only the corrective downtime of the system.

$$A_I = \frac{MTTF}{MTTF + MTTR} \text{ OR } A_I = \frac{MTBF}{MTBF + MTTR}$$

Achieved Availability, is very similar to inherent availability with the exception that preventive maintenance (PM) downtimes are also included. Specifically, it is the steady state availability when considering corrective and preventive downtime of the system. It can be computed by looking at the mean time between maintenance actions, *MTBM* and the mean maintenance downtime,

$$A_A = \frac{MTBM}{MTBM + \overline{M}}$$

Operational Availability is a measure of the average availability over a period of time and it includes all experienced sources of downtime, such as administrative downtime, logistic downtime, etc. Operational availability is the ratio of the system uptime and total time. Mathematically, it is given by:

$$A_o = \frac{Uptime}{Operating\ Cycle}$$

Where the operating cycle is the overall time period of operation being investigated and uptime is the total time the system was functioning during the operating cycle.

(*Note*: The operational availability is a function of time, t, or operating cycle.) When there is no specified logistic downtime or preventive maintenance, it returns the Mean Availability of the system. The operational availability is the availability that the customer actually experiences. It is essentially *a posterior* availability based on actual events that happened to the system. The previous availability definitions are *a priori* estimations based on models of the system failure and downtime distributions. In many cases, operational availability cannot be controlled by the manufacturer due to

variation in location, resources and other factors that are the sole province of the end user of the product.

Cost functions are often used to evaluate the performance of a RS. The form of this function depends on the reliability and maintainability characteristics of the RS of interest. However, these functions typically include a subset of the following cost parameters.

- c_f cost of a failure
- c_d cost per time unit of "downtime"
- c_r cost (per time unit) of Corrective Maintenance
- c_p cost (per time unit) of Preventive Maintenance
- c_a cost of RS replacement

9.3. RS Models and Availability

9.3.1. Renewal models

The first class of RS models that we address is based on concepts and results from renewal theory. For a repairable system, the time of operation is not continuous. In other words, its life cycle can be described by a sequence of up and down states. The system operates until it fails, then it is repaired and returned to its original operating state. It will fail again after some random time of operation, get repaired again, and this process of failure and repair will repeat. This is called a renewal process and is defined as a sequence of independent and non-negative random variables. In this case, the random variables are the times-to-failure and the times-to-repair/restore. Each time a unit fails and is restored to working order, a renewal is said to have occurred. This type of renewal process is known as an *alternating renewal process* because the state of the component alternates between a functioning state and a repair state. A system's renewal process is determined by the renewal processes of its components.

9.3.1.1. System Structure and Assumptions

We first consider a RS that is modeled as a single component or a "black box". For this RS, the duration of an interval of function is a random variable. Upon failure, CM is performed and restores the RS to a "good as new" condition. The durations of successive CM intervals are assumed to be independent and identically distributed

random variables. No PM is performed. Note that the "good as new" assumption is the key assumption and often the subject of criticism of the corresponding models (except when CM corresponds to RS replacement). Let T_i denote the duration of the i_{th} interval of RS function. Because of the "good as new" assumption, $\{T_1, T_2, \dots\}$ is a sequence of iid random variables. Let D_i denote the duration of the i_{th} CM action. Recall that $\{D_1, D_2, \dots\}$ are assumed to be iid random variables. Therefore, each cycle (function, CM) has identical probabilistic behavior, and the completion of a CM action is a renewal point for the stochastic process $\{X(t), t \geq 0\}$.

9.3.1.2. General Results

Regardless of the probability distributions governing T_i and D_i, the limiting availability is easy to obtain:

$$A = \frac{E(T_i)}{E(T_i) + E(D_i)} = \frac{MTTF}{MTTF + MTTR}$$

Suppose T_i is a Weibull random variable having shape parameter $\beta = 2$ and scale parameter $\eta = 1000$ hours. Then

$$R(t) = \Pr(T_i > t) = \exp\left[-\left(\frac{t}{\eta}\right)^{\beta}\right]$$

$$MTTF = \eta\,\Gamma\left(1 + \frac{1}{\beta}\right) = 886.2 \text{ hours}$$

Suppose D_i is a normal random variable having a mean ($MTTR$) of 25 hours. Thus,

$$A = \frac{886.2}{886.2 + 25} = 0.9726$$

For this example, availability and average availability values can be estimated using simulation.

9.3.1.3. Special Case

Suppose T_i is an exponential random variable having failure rate λ, and D_i is an exponential random variable having repair rate μ. Then

$$A(t) = \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t}$$

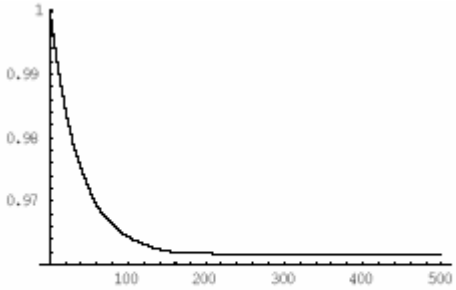
$$A = \frac{\mu}{\lambda+\mu}$$

$$A_{avg}(T) = \frac{\lambda\left[1 - e^{-(\lambda+\mu)T}\right] + \mu(\lambda+\mu)T}{(\lambda+\mu)^2 T}$$

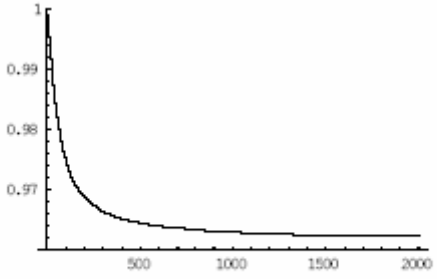
For example, suppose λ = 0.001 failures per hour (*MTTF* = 1000 hours) and μ = 0.025 repairs per hour (*MTTR* = 40 hours). In this case,

$$A = \frac{0.025}{0.001+0.025} = \frac{1000}{1000+40} = 0.9615$$

A plot of $A(t)$ and a plot of $A_{avg}(T)$ can be found in Figure below:



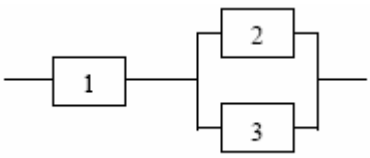
Example Availability Function



Example Average Availability Function

9.3.1.4. System Availability

Suppose a RS is comprised of independent components and we model failure/CM at the component level. In such cases, we can use component (limiting) availabilities to compute (sub)system (limiting) availability just as we do with component/(sub)system reliability. (Sub)system average availability functions must be obtained by integrating the (sub)system availability function. For a simple example, consider a 3-component system that can be described using the reliability block diagram found in Figure below



Let λ_j denote the failure rate of component *j*, let μ_j denote the repair rate of component *j*, let $A_j(t)$ denote the availability function for component *j*, and let A_j denote the limiting availability of component *j*. Then,

$$A_j(t) = \frac{\mu_j}{\lambda_j + \mu_j} + \frac{\lambda_j}{\lambda_j + \mu_j} e^{-(\lambda_j + \mu_j)t}$$

$$A_j = \frac{\mu_j}{\lambda_j + \mu_j}$$

Let $A_{23}(t)$ denote the availability function for the subsystem comprised of components 2 and 3, and let A_{23} denote the corresponding limiting availability. Then

$$A_{23}(t) = 1 - (1 - A_2(t))(1 - A_3(t))$$

$$A_{23} = 1 - (1 - A_2)(1 - A_3)$$

Let $A(t)$ denote the availability function for the RS, and let A denote the corresponding limiting availability. Then,

$$A(t) = A_1(t)A_{23}(t)$$

$$A = A_1 A_{23}$$

Component, subsystem and system average availability functions must be obtained by integrating the corresponding availability function.

9.3.2. Minimal Repair Models

The second class of RS models that we address is based on the concept of minimal repair. It is frequently the case that repair consists of replacing or restoring a parts or components leaving the approximately the same state(age) as it was in just prior to failure. This implies that time between failure may not longer be independent and identically distributed.

9.3.2.1. System Structure and Assumptions

Again, we consider a RS that is modeled as a single component or a "black box". For this RS, the duration of an interval of function is a random variable. Upon failure, instantaneous CM is performed (no PM is performed). CM restores the RS to a "bad as old" condition, i.e. the RS functions after repair but its equivalent age is the same as it was at the time of failure. For this reason, such CM is referred to as *minimal*

repair. As with the "good as new" assumption, the realism of the "bad as old" assumption is often questioned.

9.3.2.2. General Results

Let T denote the duration of the first interval of RS function. Let $f(t)$ denote the pdf of T, let $F(t)$ denote the cdf of T, and let $z(t)$ denote the hazard function of T. If T is an exponential random variable having constant failure rate λ, then $\{N(t), t \geq 0\}$ is a Poisson process having rate λ. In this case, minimal repair is equivalent to renewal. Otherwise, $\{N(t), t \geq 0\}$ is a non-homogeneous Poisson process (NHPP) having intensity function $z(t)$ or m(t).

Poisson Process

If $\{N(t), t \geq 0\}$, number of failure in (0,t] is a Poisson process having rate λ, then $N(t)$ is a Poisson random variable with mean λt, and

$$N(t) \in \{0,1,\ldots\}$$

$$E[N(t)] = Var[N(t)] = \lambda t$$

$$\Pr[N(t) = n] = \frac{e^{-\lambda t}(\lambda t)^n}{n!}$$

Furthermore, $N(t+s) - N(s)$, the number of failures in the interval $(s,t+s]$, is also a Poisson random variable having mean λt . The implication of this result is that the number of failures in a given interval depends only on the length of the interval. Note that this is not true for an NHPP.

NHPP

If $\{N(t), t \geq 0\}$ is a non-homogeneous Poisson process having intensity function $z(t)$, then $N(t)$ is a Poisson random variable having mean $Z(t)$, where $Z(t)$ is the cumulative intensity function..

$$Z(t) = \int_0^t z(u)\,du$$

Furthermore, $N(t+s) - N(s)$ is a Poisson random variable having mean $Z(t+s) - Z(s)$

$$N(t) \in \{0,1,\ldots\}$$

$$E[N(t)] = Var[N(t)] = Z(t)$$

$$\Pr[N(t) = n] = \frac{e^{-Z(t)}[Z(t)]^n}{n!}$$

$$N(t+s) - N(s) \in \{0,1,\ldots\}$$

$$E[N(t+s) - N(s)] = Z(t+s) - Z(s)$$

$$Var[N(t+s) - N(s)] = Z(t+s) - Z(s)$$

$$\Pr[N(t+s) - N(s) = n] = \frac{e^{-[Z(t+s)-Z(s)]}[Z(t+s) - Z(s)]^n}{n!}$$

For example, suppose T is a Weibull random variable having shape parameter β and scale parameter η. Then

$$z(t) = \frac{\beta}{\eta^\beta} t^{\beta-1}$$

$$Z(t) = \left(\frac{t}{\eta}\right)^\beta$$

If $\beta > 1$ ($\beta < 1$), then the intensity function increases (decreases) and failures tend to occur more (less) frequently over time. Suppose $\beta = 1.75$ and $\eta = 1500$ hours. Then,

$$E[N(1000)] = 0.4919$$
$$E[N(2000) - N(1000)] = 1.1626$$
$$E[N(3000) - N(2000)] = 1.7092$$

$$\Pr[N(1000) > 2] = 0.0138$$
$$\Pr[N(2000) - N(1000) > 2] = 0.1125$$
$$\Pr[N(3000) - N(2000) > 2] = 0.2452$$

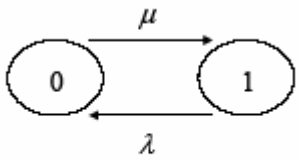
9.3.3. CTMC Models

The final class of RS models that we address is based on continuous-time Markov chains.

9.3.3.1. Single Machine Problems

Consider a single repairable machine. Let *Ti* denote the duration of the *i*th interval of machine function, and assume {*T*1, *T*2, …} is a sequence of iid exponential random variables having failure rate λ. Upon failure, the machine is repaired. Let *Di* denote the duration of the *i*th machine repair, and assume {*D*1, *D*2, …} is a sequence of iid exponential random variables having repair rate μ.. No PM is performed on the

machine. Recall that $X(t)$ denotes the state of the machine at time t. Under these assumptions, $\{X(t), t \geq 0\}$ transitions among two states, and the time between transitions is exponentially distributed. Thus, $\{X(t), t \geq 0\}$ is a CTMC having the rate diagram shown in Figure below.



Single Machine Rate Diagram

We can easily analyze the "steady-state" behavior of the CTMC. Let ϱj denote the long-run probability that the CTMC is in state j. We use balance equations to identify these probabilities. Each state of the CTMC has a balance equation that corresponds to the identity "rate in" = "rate out". For the rate diagram in Figure, the balance equations are:

$$\text{state 0: } \lambda\rho_1 = \mu\rho_0$$

$$\text{state 1: } \mu\rho_0 = \lambda\rho_1$$

These balance equations are equivalent, so we need an additional equation to solve for $\varrho 0$ and $\varrho 1$. We use the fact that the steady-state probabilities must sum to one.

$$\rho_0 + \rho_1 = 1$$

We then use the two equations to solve for the two unknowns.

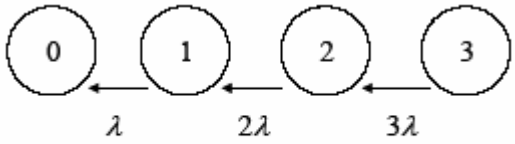
$$\rho_1 = \frac{\mu}{\lambda + \mu}$$

$$\rho_0 = \frac{\lambda}{\lambda + \mu}$$

9.3.3.2. Multiple Machine Problems

Suppose the repairable "system" of interest actually consists of m identical machines that correspond to the assumptions of the previous section. To model this situation using a CTMC, we must first modify our definition of the system state $X(t)$. Let $X(t)$ now represent the number of machines functioning at time t. However, $\{X(t), t \geq 0\}$ is still a CTMC because the number of states is discrete and transition times are

exponentially distributed. A partial rate diagram for the case in which m = 3 is constructed in Figure below.



Partial Rate Diagram when $m = 3$

Note that the repair rates on the diagram depend on *s*, the number of maintenance technicians in the system. Note that we assume each repair requires exactly one technician. Suppose $m = 3$, $s = 2$, $\lambda = 1$ failure per day, and $\mu = 5$ repairs per day. The completed rate diagram for the resulting CTMC is given in next Figure. Note that the transition rate from state 3 to state 2 is 3. This is because 3 machines are functioning; each has a failure rate of 1, so the total failure rate is that the transition rate from state 1 to state 2 is 10. This is because 2 machines are failed; this implies that both technicians are repairing at a rate of 5, so the total repair rate is 10. The Balance equations:

$$\text{state 0: } \rho_1 = 10\rho_0$$

$$\text{state 1: } \rho_2 = 50\rho_0$$

$$\text{state 3: } \rho_3 = \frac{250}{3}\rho_0$$

Whose solution is:

$$\rho_0 = \frac{3}{433}$$

$$\rho_1 = \frac{30}{433}$$

$$\rho_2 = \frac{150}{433}$$

$$\rho_3 = \frac{250}{433}$$

Note that we can use the steady-state probabilities to obtain both machine and technician utilization. For example, the average number of machines functioning is

$$AvgM = 0\rho_0 + 1\rho_1 + 2\rho_2 + 3\rho_3 = 2.49 \text{ (83\% utilization)}$$

and the average number of busy technicians is

$$AvgS = 2\rho_0 + 2\rho_1 + 1\rho_2 + 0\rho_3 = 0.50 \text{ (25\% utilization)}$$

At this point, a reasonable question is: How many technicians should be assigned to maintain these machines, i.e. should s = 1, 2 or 3? To answer this question, first we modify the CTMC for the cases in which s = 1 and s = 3. Then, we compute the steady-state probabilities and utilization measures for each case. Then, we can use an economic model to determine the optimal value of s. Let cs denote the cost per day of employing a technician, let cd denote the cost per day of machine downtime, and let C denote the cost per day of system operation. Then, expected cost:

$$E(C) = c_s s + c_d(m - AvgM)$$

An interesting variation of the multiple machine problem is the case in which the machines are not identical. For example, suppose a system contains two machines of different types that are repaired upon failure (no PM), and suppose two equally trained technicians maintain these machines. Let λi denote the failure rate for machine i, and let $\mu\ i$ denote the repair rate for machine i. Modeling this problem using a CTMC requires a more complex definition of the system state.

$$X(t) = \begin{cases} 1,1 & \text{both machines are functioning} \\ 1,0 & \text{machine 1 is functioning, machine 2 is down} \\ 0,1 & \text{machine 1 is down, machine 2 is functioning} \\ 0,0 & \text{both machines are down} \end{cases}$$

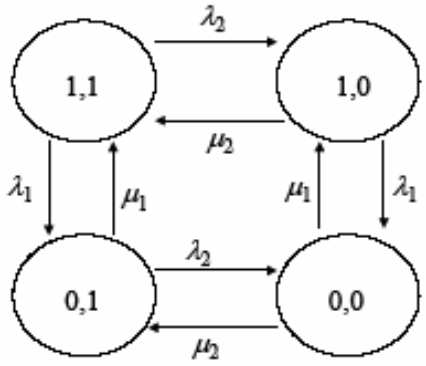
The corresponding rate diagram is provided in figure below. For example, suppose $\lambda 1$ = 1, $\mu 1$ = 8, $\lambda 2$ = 2 and μ 2 =10. Construction and solution of the balance equations yields $\varrho 1,1$ = 0.7407, $\varrho 1,0$ = 0.1481, $\varrho 0,1$ = 0.0926 and $\varrho 0,0$ = 0.0185. The steady-state probabilities can then be used to compute machine availability

machine 1: $\rho_{1,1} + \rho_{1,0} = 0.8889$
machine 2: $\rho_{1,1} + \rho_{0,1} = 0.8333$

and machine and technician utilization

$AvgM = 2\rho_{1,1} + \rho_{1,0} + \rho_{0,1} = 1.7222$ (86%)

$AvgS = \rho_{1,0} + \rho_{0,1} + 2\rho_{0,0} = 0.2778$ (14%)



Rate Diagram for Two Different Machines

9.4. Maintainability

The performance of any maintenance task is related to the associated costs both in terms of the cost of the maintenance resources and cost of the consequences of not having the system available for operation. Therefore, maintenance departments are one of the major cost centres, costing industries billions of rupees each year and as such they have become a critical factor in the profitability equation of many organization. Thus as maintenance actions are becoming increasingly costly, maintainability engineering is gaining recognition day by day. For instance, in the journal Aviation Week and Space Technology, January 1996, it was reported that by the year 2000, US Air Force would begin looking at upgrades of a heavy air lifter aircraft C-5A. The comment was *that although the structure of the aircraft is considered to be good, 'the reliability and maintainability leave a lot to be considered.'* It is inevitable that in future too considerations and comments like this will significantly increase and that the impact of these considerations on the final selection of system will be far greater.

Although it is extremely important for the operators/users to know the functionality, durability and reliability characteristics of the system at the beginning of its operational life, it is equally, or even more important for them to have information regarding issues like:

- Which maintenance task should be performed?
- When should the maintenance task be performed?
- Difficulty level of the maintenance task.
- Safety-level of the maintenance task.
- How many people are required to perform the maintenance task? Their skill-levels and expertise?
- How much is the restoration going to cost?
- How long the system is going to failure?
- What is the equipment requirement?

In most of the cases, the answers to these questions provided by the designer/manufacturer are very basic and limited. For instance, in case of a motor vehicles the answers cover no more than the list of maintenance activities which should be performed during regular service every 5000/10000 Km or so. All the above questions remain unanswered and the users are left to find the answers by themselves. The reason for this is the fact that up to now the main purpose and concern of designers has been the achievement of functionality, *whereas the ease of maintaining functionality by the users has been almost ignored. Traditionally, it was the problem of the maintenance personnel, not the designers.*

However, the situation is changing gradually, thanks to aerospace and military customers who recognized the importance of information of these types and who made it a characteristic equally desirable as performance, reliability, availability and similar.

As no scientific disciplines were available to help designers and producers to provide answer to the above questions, the need arose to form a new discipline. Maintainability theory was created- *a scientific discipline, which studies complexity, factors and resources related to the tasks needed to performed by the user in order to maintain the functionality of the product, and works out method for their quantification, assessment, prediction and improvement.* It is rapidly growing in importance because of its considerable contribution towards the reduction of maintenance cost of a system during its utilization.

Maintainability as a characteristic of man-made system MIL-STD-721C (1966), defines the maintainability as a characteristic of design and installation, which is expressed as the probability that an item will be retained in or restored to specified condition within a given period of elapsed time, when maintenance is performed in accordance with prescribed procedures and resources.

9.4.1. Maintainability Impact on Availability

The majority of users state that they need the equipment availability as badly as they need safety. There are several ways designers can control that. One is to build items/systems that are extremely reliable and consequently, costly. Second, is to provide a system that, when it fails, it is easy to restore. However, if everything is made highly reliable and everything is easy to repair, the producer has got a very efficient system, which no one can afford to buy. Consequently, the question is how much a utility of system is needed and how much one is prepared to pay for it? Consequently, maintainability is one of the factors in achieving a high level of operational availability, which in turn increases users' satisfaction.

9.4.2. Maintainability Measures

Duration of maintenance tasks can only be described in probabilistic terms and are fully defined by the RV DMT (duration of maintenance task) and its probability distribution, m(t). The most frequently used maintainability characteristics are:

1. Probability of task completion.
2. Mean duration of maintenance task.
3. Percentage duration of maintenance task.
4. Variability of duration of maintenance task.
5. Success of task completion

A brief definition and description of these characteristics are as follow:

9.4.2.1. Probability of Task Completion(PTC)

It represents the probability that the maintenance task considered will be successfully completed by a stated time, T_{st}.

$$PTC_{DMT} = P\{DMT \le T_{st}\} = \int_0^{T_{st}} m(t)dt$$

9.4.2.2. Mean Duration of Maintenance Task (MDMT)

It is denoted as E(MDMT), and represents the expectation of the RV DMT, which can be used for calculation of the characteristic of maintenance task, i.e.,

$$MDMT = E(DMT) = \int_0^{\infty} tm(t)dt = \int_0^{\infty} [1 - M(t)]dt$$

Which represent the area below the function, which is complementary of maintainability function.

9.4.2.3. Percentage Duration of Maintenance Task(DMT_p)

It represents the duration of maintenance task by which a given percentage of maintenance tasks considered will be successfully completed, i.e.,

$$DMT_p = t \rightarrow for\ which\ \mathrm{M(t)} = P(DMT \le t) = \int_0^{t} m(t)dt = p$$

The most frequently used measure of DMT_{90} is the time, which presents the restoration time by which 90% of maintenance trials will be completed, i.e.,

$$DMT_p = t \rightarrow for\ which\ \mathrm{M(t)} = P(DMT \le t) = \int_0^{t} m(t)dt = 0.9$$

It is worth noticing that in military-oriented literature and defense contracts, the numerical value of DMT_{95} to as maximum repair time and is denoted as M_{max}, thus $M_{max} = DMT_{95}$.

9.4.2.4. Variability of Duration of Maintenance Task (CV(DMT))

In certain cases it is difficult, when using only the knowledge of a standard deviation to decide whether the dispersion is particularly large or small, because this will depend on the mean value. In these, situation coefficient of variation (CV) is defined as:

$CV(DMT) = \frac{SD(MDMT)}{MDMT}$ is very useful because it provides better information regarding the dispersion, also known as the variability of the RV, in general terminology.

9.4.2.5. Success of Task Completion (STC)

It represents the probability that the trial, which has not been completed at time t_1 will be finished by the time, t_2 (example of conditional probability, i.e., the task could be completed by t_2, given that it was not completed at t_1).

$$STC(t_1, t_2) = P(DMT \le t_2 \mid DMT > t_1) = \frac{M(t_2) - M(t_1)}{1 - M(t_1)}$$

with M(0)=0.

This measure of maintenance provides very useful information for the planners and managers.

The maintenance measures defined so far relate to the duration of maintenance task expressed through the probability distribution of the elapsed maintenance times. Besides the elapsed time, one must also consider the demand for resources required for the task execution, in particular the number of personnel involved. In some instances, the elapsed time can be reduced by employing additional personnel. However, this may turn out to be an expensive trade-off, particularly when high skill levels are required. One of the most frequently used maintainability measure is MMPD.

9.4.2.6. Maintenance Personnel Demand per Maintenance Task (MMPD)

It is quantified and represented by the following expression:

$$MMPD^{MT} = \frac{\sum_{i=1}^{nma} MPSD_i \times MDMA_i}{MDMT}$$

where, *MPSDi* for the maintenance personnel demand for the successful completion of ith maintenance activity, *MDMAi* is the mean duration of the ith maintenance activity, and *nma* represents the total number of activities, which make up the task under consideration.

Example:

For the maintenance task, whose duration time could be modeled by the Weibull distribution with parameters, characteristic life = 29 minutes and shape parameter = 2.9,

1. The probability that the task analyzed would be completed within 20 minutes = 0.29.
2. The duration time up to which 20 % and 90 % of task will be completed, will be 17.29 minutes and 42.33 minutes, respectively
3. The mean duration of maintenance task, MDMT = 25.87 minutes
4. The probability that the maintenance task, which has not been completed during the first 29 minutes will be completed within the following 10 minutes = 74.5%

Example:

The maintenance task under consideration consists of four different maintenance activities:

Activity	Mean Duration (min)	Number of Personnel
1	30	1
2	120	3
3	45	1
4	5	2

The maintenance personnel demand = 2.225 (Just over 2)

The above maintainability measures described are related to the single maintenance task. However, there are large number of items the maintenance of which requires two or more different nature of maintenance tasks such as:

- Corrective in nature in response to different failure modes in whish it can fail
- Preventive, where the maintenance tasks are performed in order to reduce the probability of occurrence of the failure due to a specific failure mechanisms (corrosion, fatigue, wear, thermal deformation etc…)
- Conditional nature, where the tasks are performed in order to assess the condition of the item in order to determine the further course of action.

Therefore, an item exposed to several different maintenance tasks, it is necessary to determine the maintainability measures in such a way that all task tasks are somehow taken into consideration during the stated operational length.

9.4.3. Item Based Statistics

9.4.3.1. Mean Time in Maintenance (MTIM)

The mean time in maintenance during a stated operational length, Lst is calculated as:

$$\boldsymbol{MTIM(L_{st}) = \sum_{i=1}^{nte} MDMT_i \times MNMT_i(L_{st})}$$, where nte, number of different maintenance tasks expected to be performed on the items (Output of FMEA), MNMTi(Lst), mean number of tasks expected to be performed during the stated length of period. Thus,

$$MTIM^c(L_{st}) = \sum_{i=1}^{nct} MDMT^c{}_i \times MNMT^c{}_i(L_{st})$$ (Time in Corrective Maintenance)

$$MTIM^p(L_{st}) = \sum_{i=1}^{npt} MDMT^p{}_i \times MNMT^p{}_i(L_{st})$$ (Time in Preventive Maintenance)

$$MTIM^m(L_{st}) = \sum_{i=1}^{nmt} MDMT^m{}_i \times MNMT^m{}_i(L_{st})$$ (Time in Conditional Maintenance)

The most frequently used maintainability statistics of merit for an item are:

9.4.3.2. Mean Time to Restore (MTTR)

It represents the mean duration of the maintenance task required to restore the functionality, when two or more different tasks could be demanded.

$$\boldsymbol{MTTR(L_{st}) = \frac{MTIM(L_{st})}{MNMT(L_{st})}}$$ (Equation remains the same for corrective, preventive or conditional except a suffix can be placed as in for MTIM equations)

9.4.3.3. Maintenance Hours per Operational Unit (MHOU)

$$\boldsymbol{MHOU(L_{st}) = \frac{MTIM(L_{st})}{L_{st}}}$$, where operational unit could be hours, kilometers, landings, cycles, week etc…

9.4.4. System Based Statistics

It is determined according to the number of consisting items as well as the number of maintenance tasks associated with each of them.

$$MTTR_s = \frac{\sum_{i=1}^{nmi} MTIM_i(L_{st})}{\sum_{i=1}^{nmi} MNMT_i(L_{st})}$$, where nmi, number of maintenance-significant items within the system, MTIM is the mean time in maintenance of the ith item

In short, the statistics or measures used in general or on item can easily be extended to the system levels.

9.4.5. Other Areas of Maintainability Engineering

- Maintainability Allocation
- Prediction of Maintainability Measures
- Maintainability Management
- Maintainability Demonstration Tests

9.5. Maintenance and Optimization

Maintenance- what is it?

- Actions associated with equipment when it breaks.
- Work of keeping something in proper conditions; upkeep.

Definition : Maintenance is the actions taken to prevent a device or component failing or to repair normal equipment degradation experienced with the operation of the device to keep it in proper conditions.

In other words we try to keep non-failed device to their operating conditions with respect to reliability and safety and if they have failed, we try to restore them to the operating state preferably without interrupting the system operation.

Objectives of a good maintenance programme could be-

1. to provide the freedom free breakdown during operation.
2. to maintain equipment in satisfactory condition for its safe, sound and proper operation.
3. to maintain equipment at its maximum operating frequency and efficiency.

4. keep the equipment downtime to its minimum from any breakdown or shutdown.

in order to minimize the maintenance cost to its minimum

To achieve the objectives of a maintenance programme, one has to adopt certain optimal or efficient maintenance strategy, which are concerned with directing the resources where the strategy adopted may be influenced by-

- Production requirement
- System conditions & age
- Internal / external resources
- Safety considerations
- Other statutory regulations.

Maintenance Classification:

1. Reactive maintenance
2. Proactive maintenance
    - Preventive maintenance
    - Predictive maintenance
    - RCM (Reliability Centered Maintenance)

9.5.1. Reactive Maintenance

- Operate the system until it breaks- scrap it, buy new one.
- Operate the system until it breaks- repair it.
- Operate the system until it breaks and then shell it before it breaks down or overhaul / repair becomes too costly.

Advantages:

1. No associate maintenance cost .
2. No manpower requirement.
3. No labour cost.

Disadvantages:

1. Shortening the life of the component.
2. Frequent replacement of the component.
3. Damage of the secondary equipment from equipment failure.
4. The repair or labour cost expected to high because the failure may require an extensive repair.
5. Increase cost due to unplanted downtime equipment.

Proactive maintenance: In this category we carry out regular maintenance such as inspection, repair, or replacement lubricating adjustment, alignment, cleaning etc which are plant careful in conjunction with protection requirement to prevent failure of the equipment during its normal operations.

Preventive Maintenance : These are the actions performed on a time or machine run based scheduled that detect preclude or mitigate the degradation of the system / component with the aim of sustaining its useful life through controlling degradation to an expectable level.

Advantages:

1. Increased component life.
2. Decreased process / equipment failure.
3. Saving in energy.
4. Estimated 12-18% in cost saving over Reactive Maintenance.

Disadvantages:

1. Catastrophic failure cannot be avoided.
2. Since it is time based, potential for incidental damage to component in conducting unneeded maintenance.
3. Labour intensive.

9.5.2. Predictive Maintenance

It is done based on quantifying material or equipment conditions that can detect the onset of degradation mechanism. Thereby allowing casual stresses to be eliminated or

controlled prior to any significance detoriation in the component or equipment in physically state.

Advantages:

1. Increased component operation life and availability.
2. Decreased component or equipment downtime.
3. Decreased cost of parts and labour.
4. Improved workers moral.
5. Better quality product.
6. Saving in energy.
7. Estimated that 8-12% saving over Preventive Maintenance programme.

Disadvantages:

1. Investment on diagnostic equipment.
2. Personnel training.
3. Management does not readily see its saving potential.

In order to keep the equipment under healthy and operable conditions, an organization has to take many decisions such as:

- Replacement Decisions
- Inspection Decisions
- Overhaul and Repair Decisions.
- Organizational Structure Decisions (What Facilities, i.e., manpower and equipment Or How these facilities to be used taking into account the possible use of subcontractors etc…)
- Reliability Decisions.

In this context, we would describe some basic models.

9.5.3. Replacement Decision

Assumptions:

1. State of the system should be known (working or failed)

2. Total cost of replacement is high after the fail.
3. Replacement action returns the system as good as new.
4. The failure rate of the system is IFR.

Some Replacement Polices:

1. Optimal replacement times for equipment whose operating cost increases with its use.

   Objective: Make a balance between the money spent on replacements and savings obtained by reducing the operating cost.

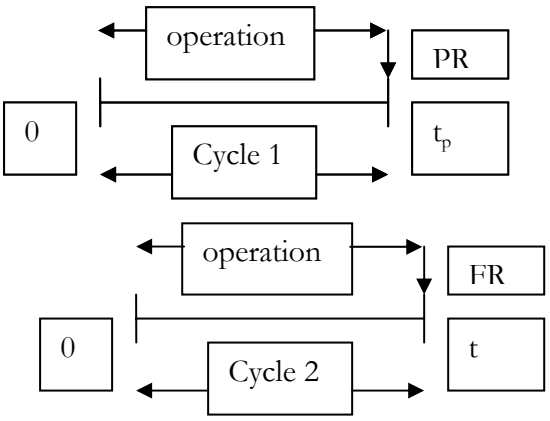
2. Optimal replacement policy for equipment whose operating cost increases with use: finite time horizon.

   Objective: Determine an optimal replacement policy (i.e. sequence of decisions) which tells us, when equipment reaches a particular age, whether or not it should be replaced or continue to be operated to minimize the total cost of operation and replacement over a fixed future period of time.

3. Optimal replacement policy for two machines one of which acts as a standby, when the operating cost of a machine increases with use.

4. Optimal interval between preventive replacements of equipment subject to breakdown.

   Objective: The replacement policy is one where preventive replacements occur at fixed intervals of time, and failure replacements occur as and when necessary, and we want to determine the optimal interval between the preventive replacements to minimize the total expected cost of replacing the equipment per unit time.



The total expected cost per unit time, for preventive replacement at time t_p =C (t_p) = [total expected cost in interval (0,t_p)] /[Length of interval]

Total expected cost = cost of preventive replacement in interval (0,t_p)

+ expected cost of failure replacement. = $C_p + C_f.H(t_p)$

C_p = cost of PR, C_f =cost of FR

Where $H(t_p)$ = expected number of failures in interval (0,t_p]

$$C(t_p) = [C_p + C_f.H(t_p)] / t_p$$

5. Optimal preventive replacement age of equipment subject to breakdown.
Objective: To determine the optimal replacement age of the equipment to minimize the total expected replacement cost per unit time,

There are two type of operation-

Total expected replacement cost per unit time =C(t_p) = [Total expected replacement cost per cycle] / [expected cycle length]

Total expected replacement cost per cycle = cost of a preventive cycle × probability of preventive cycle + cost of a failure cycle × probability of failure cycle = $C_{p.} R(t_p) + C_{f.}[1- R(t_p)]$

Expected cycle length = length of a preventive cycle × probability of preventive cycle + expected length of a failure cycle × probability of failure cycle = $t_{p.}R(t_p) + M(t_p).[1- R(t_p)]$

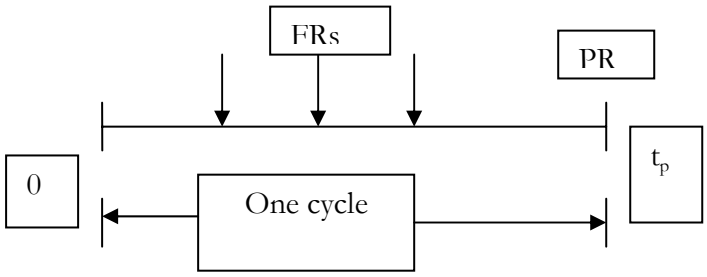
Where $M(t_p) = \int_{-\alpha}^{t_p} \frac{tf(t)dt}{1-R(t_p)}$ = expected length of a failure cycle

$$C(t_p) = \frac{C_p.R(t_p) + C_f.[1-R(t_p)]}{t_p.R(t_p) + M(t_p).[1-R(t_p)]}$$

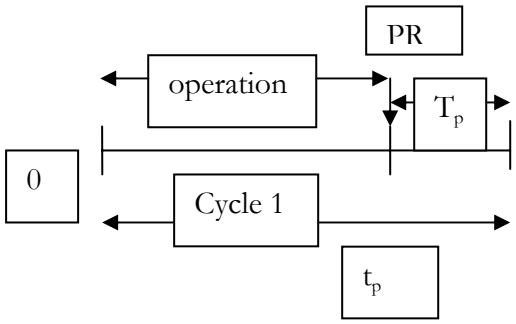
$$C(t_p) = \frac{C_p.R(t_p) + C_f.[1-R(t_p)]}{t_p.R(t_p) + \int_{-\alpha}^{t_p} \frac{tf(t)dt}{1-R(t_p)}}$$

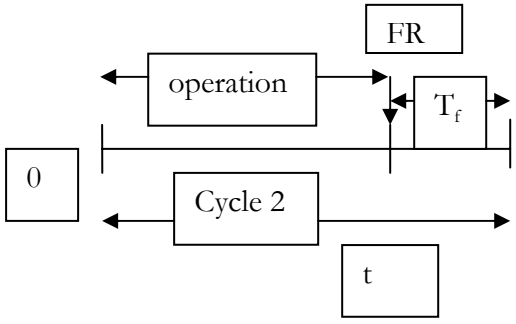
6. Optimal preventive replacement age of equipment subject to breakdown, taking account of the times required to effect to failure and preventive replacements.

Objective: The problem is identical of the previous problem except that,



instead of assuming that the failure and failure replacement are made instantaneously, account is taken of the time required to make these replacements.





T_p = Time required to PR

T_f = Time required to FR

Total expected replacement cost per unit time =$C(t_p)$ = [Total expected replacement cost per cycle] / [expected cycle length]

Total expected replacement cost per cycle = cost of a preventive cycle × probability of preventive cycle + cost of a failure cycle × probability of failure cycle = $C_{p.}\, R(t_p) + C_{f.}[1- R(t_p)]$

Expected cycle length = length of a preventive cycle × probability of preventive cycle + expected length of a failure cycle × probability of failure cycle = $(t_p + T_p).R(t_p) +[M(t_p) + T_f].[1- R(t_p)]$

$$C(t_p) = \frac{C_p.R(t_p) + C_f.[1 - R(t_p)]}{(t_p + T_p).R(t_p) + [M(t_p) + T_f].[1 - R(t_p)]}$$

7. Optimal preventive replacement interval or age of equipment subject to breakdown: minimization of downtime.

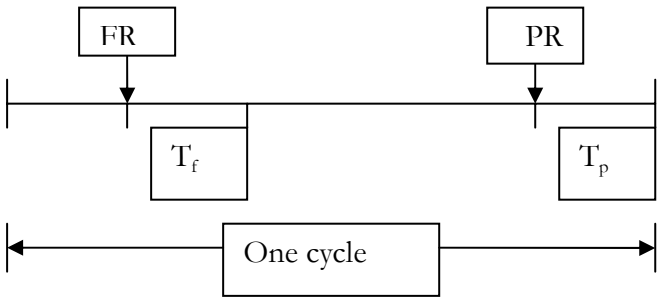
   Objective:- Optimal PR interval or Age of equipment subject to breakdown: Minimization of downtime(age is not considered)

   The total down time per unit time, for PR at time t_p - $D(t_p)$

   $D(t_p)$ = (Expected down time due to failure + down time due to PR)

   / (Cycle Length)

   Down time due to failure = Number of Failure in the interval(0, t_p) × Time required to make a FR = $H(t_p) \times T_f$

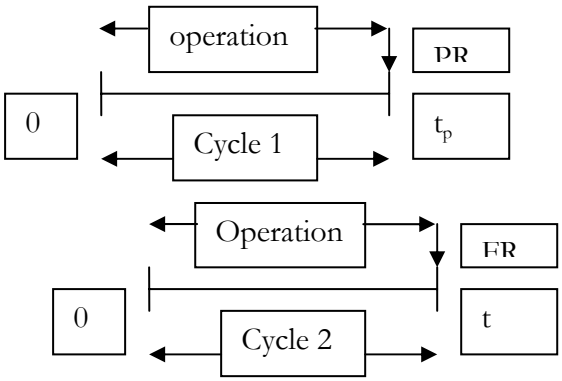


$$D(t_p) = \frac{H(t_p) \times T_f + T_p}{t_p + T_p}$$

8. Optimal preventive replacement interval or age of equipment subject to breakdown: minimization of downtime, taking account of the times required to effect to failure and preventive replacements.

Objective: To determine the objective length at which PRs should occur such that total down time per unit time is minimized.(Age is considered)

The total down time per unit for PRs once the equipment becomes of age $t_p =$ $D(t_p) =$ (Total Expected down time per cycle)/(Excepted Cycle



Length)

Total expected down time/cycle= Down time due to preventive cycle × Probability of preventive cycle + down time due to FR × probability of failure cycle = $T_p.R(t_p) + T_f.[1-R(t_p)]$

Expected cycle length = $(t_p+T_p).R(t_p)+[M(t_p)+T_f].[1-R(t_p]$ (same as in model 6)

$$D(t) = \frac{T_p.R(t_p) + T_f.[1 - R(t_p)]}{(t_p + T_p).R(t_p) + [M(t_p) + T_f].[1 - R(t_p)]}$$

9.5.4. Inspection Decisions (Inspection Models)

9.5.4.1. Optimal inspection frequency: Maximization of profit.

Objective:

Determine an inspection policy, which will give us a balance between the number of inspections and the resulting output such that the profit per unit time from the equipment is maximized over a long period.

Profit per unit time=P(n) = Value of output per uninterrupted unit of time – output lost due to repairs per unit time - output lost due to inspections per unit time – cost of repairs per unit time – cost of inspections per unit time.

Output lost due to repairs per unit time = Value of output per uninterrupted unit of time× number of repairs per unit time × Mean time to repair = $V.\lambda(n)/\mu$

Output lost due to inspections per unit time = Value of output per uninterrupted unit of time× number of inspections per unit time × Mean time to inspection = $V.n/\mu$

Cost of repairs per unit time = Cost of repairs per uninterrupted unit of time× number of repairs per unit time × Mean time to repair = $R.[\lambda(n)/\mu]$

Cost of inspections per unit time = Cost of inspections per uninterrupted unit of time× number of inspections per unit time × Mean time to inspection =I. (n/i)

$$P(n) = V - \frac{V.\lambda(n)}{\mu} - \frac{N.n}{i} - \frac{R.\lambda(n)}{\mu} - \frac{I.n}{i}$$

Where,

λ = mean arrival rate of failures

μ = mean repair rate

n = number of inspections per unit time

i = inspection times

V = profit value per uninterrupted unit of time

I = the average cost of inspection per uninterrupted unit of

R = the average cost of repair per uninterrupted unit of time

9.5.4.2. Optimal inspection frequency: Minimization of downtime.

Objective:

The problem is similar of the previous model but we have to choose 'n' to minimize total downtime per unit time.

Total down time per unit time = D(n) = Downtime incurred due to repair per unit time + Downtime incurred due to inspection per unit time = $\lambda(n)/\mu + n/i$

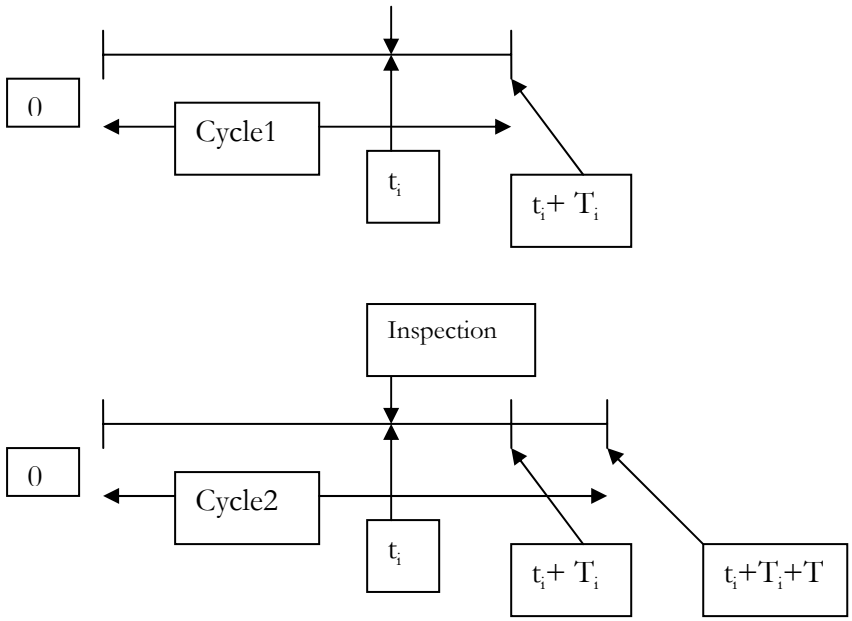
$$D(n) = \frac{\lambda(n)}{\mu} + \frac{n}{i}$$

9.5.4.3. Optimal inspection interval to maximize the availability of equipment used in emergency conditions

Objective:

To determine the interval 't_i' between inspections to maximize availability per unit time.

There are two possible cycles of operation –



Availability per unit time = $A(t_i)$ = (Expected availability per cycle) / Expected cycle length

Expected cycle length = $(t_i + T_i).R(t_i) + (t_i + T_i + T_i).[1 - R(t_i)]$

Expected availability per cycle = $t_i.R(t_i) + \frac{\int_{-\alpha}^{t_i} t.f(t)dt}{1 - R(t_i)}$.

$$= t_i.R(t_i) + \int_{-\alpha}^{t_i} t.f(t)dt$$

$$A(t_i) = \frac{t_i.T_i + \int_{-\alpha}^{t_i} t.f(t)dt}{(t_i + T_i).R(t_i) + (t_i + T_i + T_r).[1 - R(t_i)]}$$

10. Software Reliability Concepts

10.1. Terminologies

Dependable Systems: Systems that must be dependable. Dependable Systems are systems which have critical non-functional requirements for reliability, safety or security.

Software Reliability: Probability that the program performs successfully, according to specifications for a given time period.

$$R_{sy} = R_s * R_h * R_o$$

R_{sy} = System Reliability

R_s = Software Reliability

R_h = Hardware Reliability

R_o = Operator Reliability

This assumes Hardware, Software and Operator errors to be mutually exclusive.

Failure is a departure of system behavior in execution from user requirements; it is a user-oriented concept. A software failure must occur during execution of a program. Potential failures found by programmers as the result of design inspections, code reading and other methods do not count as a failure.

Fault is the defect that potentially causes the failure when executed; it is a developer-oriented concept. A software fault is a defect in code. It is caused by an error, which is an incorrect or missing action by a person or persons.

Errors are human mistakes that get into the software.

Defects are improper program conditions that generally result in an error.

Failure Intensity is an alternative way of expressing reliability. Reliability is the probability that a system will operate without failure for a specified number of natural units or a specified time known as mission time. Failure intensity is defined as failures

per unit time. Time is generally execution time or it can be natural units. This term is used in software reliability engineering because of its simplicity and intuitive appeal.

Failure severity class is a set of failures that have the same per-failure impact on users. Severity classes are assigned to failures primarily for use with failure frequencies to prioritize failures for resolution. Common classification criteria include human life, cost and system capability impacts. Each of these criteria can include many sub criteria, some of which may be important for a particular application. For example, cost impact may include extra operational cost, repair and recovery cost, and loss of present or potential business. System capability may include such sub criteria as loss of critical data, recoverability and downtime. For systems' where availability is important, failures that result in greater downtime will often be placed in a higher failure severity class.

Problems are user-encountered difficulties. They may result from failures or misuse.

Performance specification is a written requirement, figure and figure of merit, or parameter, which qualitatively or quantitatively define system performance.

Implied-specification is an unwritten requirement that is understood by the majority of the project team to be essentially equivalent to a written requirement.

Verification is an attempt to find errors by executing a program in a test or simulated environment.

Validation is an attempt to find errors by executing a program in a given real environment.

Certification is an authoritative endorsement of the correctness of a program.

10.2. Overview of Software Reliability

The IEEE defines reliability as "The ability of a system or component to perform its required functions under stated conditions for a specified period of time." To most project and software development managers, reliability is equated to correctness, that is, they look to testing and the number of "bugs" found and fixed. While finding and fixing bugs discovered in testing is necessary to assure reliability, a better way is to

develop a robust, high quality product through all of the stages of the software lifecycle. That is, the reliability of the delivered code is related to the quality of all of the processes and products of software development; the requirements documentation, the code, test plans, and testing.

Software reliability is comprised of three activities:

i. Error prevention.

ii. Fault detection and removal.

iii. Measurements to maximize reliability, specifically measures that support the first two activities.

10.2.1. Errors, Faults and Failures

The terms- errors, faults and failures are often used interchangeable, but do have different meanings. In software, an error is usually a programmer action or omission that results in a fault. A fault is a software defect that causes a failure, and a failure is the unacceptable departure of a program operation from program requirements. When measuring reliability, we are usually measuring only defects found and defects fixed. If the objective is to fully measure reliability we need to address prevention as well as investigate the development starting in the requirements phase – what the programs are developed to.

It is important to recognize that there is a difference between hardware failure rate and software failure rate. For hardware, as shown in Figure 10.1, when the component is first manufactured, the initial number of faults is high but then decreases as the faulty components are identified and removed or the components stabilize. The component then enters the useful life phase, where few, if any faults are found. As the component physically wears out, the fault rate starts to increase.

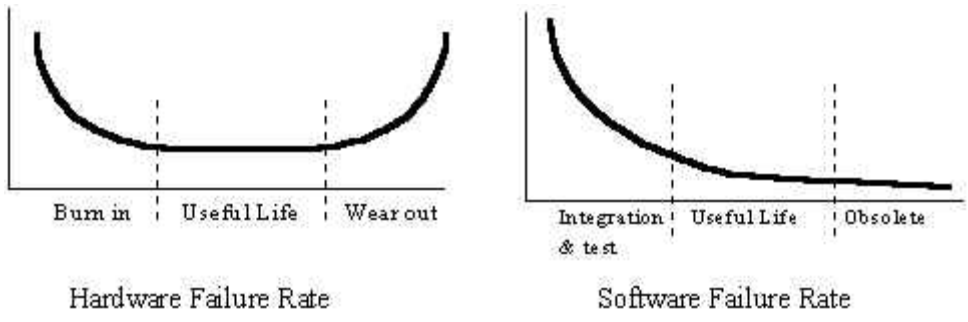


Figure 10.1: Failure Rates

Software however, has a different fault or error identification rate. For software, the error rate is at the highest level at integration and test. As it is tested, errors are identified and removed. This removal continues at a slower rate during its operational use; the number of errors continually decreasing, assuming no new errors are introduced. Software does not have moving parts and does not physically wear out as hardware, but is does outlive its usefulness and becomes obsolete.

To increase the reliability by preventing software errors, the focus must be on comprehensive requirements and a comprehensive testing plan, ensuring all requirements are tested. Focus also must be on the maintainability of the software since there will be a "useful life" phase where sustaining engineering will be needed. Therefore, to prevent software errors, we must:

i. Start with the requirements, ensuring the product developed is the one specified, that all requirements clearly and accurately specify the final product functionality.

ii. Ensure the code can easily support sustaining engineering without infusing additional errors.

iii. A comprehensive test program that verifies all functionality stated in the requirements is included.

10.2.2. Software failure mechanisms

Software failures may be due to errors, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software or other unforeseen problems. While it is tempting to draw an analogy between Software Reliability and Hardware Reliability, software and hardware have basic differences that make them different in failure mechanisms. Hardware faults are mostly *physical faults*, while software faults are *design faults*, which are harder to visualize, classify, detect, and correct. Design faults are closely related to fuzzy human factors and the design process, which we don't have a solid understanding. In hardware, design faults may also exist, but physical faults usually dominate. In software, we can hardly find a strict corresponding counterpart for "manufacturing" as hardware manufacturing process, if the simple action of uploading software modules into place does not count. Therefore, the quality of software will not change once it is uploaded into the storage and start running. Trying to achieve higher reliability by simply duplicating the same software modules will not work, because

design faults cannot be masked off by voting. A partial list of the distinct characteristics of software compared to hardware is listed below:

i. **Failure cause**: Software defects are mainly design defects.

ii. **Wear-out**: Software does not have energy related wear-out phase. Errors can occur without warning.

iii. **Repairable system concept**: Periodic restarts can help fix software problems.

iv. **Time dependency and life cycle**: Software reliability is not a function of operational time.

v. **Environmental factors**: Do not affect Software reliability, except it might affect program inputs.

vi. **Reliability prediction**: Software reliability cannot be predicted from any physical basis, since it depends completely on human factors in design.

vii. **Redundancy**: Cannot improve Software reliability if identical software components are used.

viii. **Interfaces**: Software interfaces are purely conceptual other than visual.

ix. **Failure rate motivators**: Usually not predictable from analyses of separate statements.

10.3. Software Reliability Metrics

Metric	Explanation	Example systems
POFOD (Probability of Failure on Demand)	Measure of likelihood that the system will fail when a service request is made. POFOD = 0.001 means 1 out of 1000 service requests may result in failure.	Safety-Critical and non- stop systems such as Hardware Control Systems.
ROCOF (Rate of Failure Occurrence)	Measure of frequency of occurrence with which unexpected behavior is likely to occur. If ROCOF=2/100 means 2 failures in 100 operational time units. This measure indicates failure intensity.	Operating Systems, Transaction Processing Systems
MTTF (Mean Time to Failure)	Measure of the time between observed system failures. For example, an MTTF of 500 means that 1 failure can be expected every 500 time units. It is reciprocal of ROCOF.	Systems with long transactions, such as CAD, where MTTF >Transaction time.
AVAIL (Availability)	How likely the system is to be available for use. 0.998 means in every 1000 time units, the system is likely to be available for 998 of these.	Continuously running systems such as telephone switching systems.

10.1. Measurements to assess Reliability

(a) The no. Of system failures given a number of system inputs. This is used to measure POFOD.

(b) The time (or no. of transactions) between system failures. This is used to measure ROCOF and MTTF.

(c) The elapsed repair or restart time when a system failure occurs. Given that the system must be continuously available; this is used to measure Availability.

10.2. Complimentary strategies to achieve Reliability

(a) Fault Avoidance: The design and implementation process should be organized with the objective of producing fault-free systems.

(b) Fault Tolerance: This strategy assumes that residual faults remain in the system. Facilities are provided in the software to allow operation to continue when these faults cause system failures.

(c) Fault Detection: Faults are detected before the software is put into operation. The software validation process uses static and dynamic methods to discover any faults, which remain in a system after implementation.

10.2.1. Fault Avoidance

Fault Avoidance and development of fault-free software relies on: -

(a) The availability of a precise (preferably formal) system specification which is an unambiguous description of what must be implemented.

(b) Organizational quality philosophy where-in programmers should be expected to write bug-free programs.

(c) Use of strongly typed programming language so that possible errors are detected by the language compiler.

(d) Restrictions on the use of programming constructs, such as pointers, which are inherently error-prone.

10.2.2. Fault Tolerance

A fault-tolerant system can continue in operation after some system failures have occurred. Fault tolerance is needed in situations where system failure would cause some catastrophic accident or where a loss of system operation would cause large economic losses. For example, the computers in an aircraft must continue in operation until the aircraft has landed, the computers in an air traffic control system must be continuously available. It can never be conclusively demonstrated that a

system is completely fault-free. Fault-tolerance facilities are required if the system is to be resilient to failure. There are four aspects to fault tolerance: -

(a) Failure Detection: The system must detect that a particular state combination has resulted or will result in a system failure.

(b) Damage Assessment: The parts of the system state, which have been affected by the failure, must be detected.

(c) Fault Recovery: The system must restore its state to a known safe state. This may be achieved by correcting the damaged state (Forward Error recovery) or by restoring the system to a known safe state (Backward Error recovery). Forward error recovery is more complex.

(d) Fault Repair: This involves modifying the system so that the fault does not recur. In many cases, software failures are transient and result due to peculiar combination of system inputs. No repair may be necessary as normal processing can resume immediately after fault recovery.

10.3. Error Categories

10.3.1. Design errors

This design phase of software development involves the intellectuality, creativity and intuitively of the designer. So that design phase has many errors, since are developed by human beings. Few of the approaches, which are described below, can be avoided or can be considered to avoid errors.

(a) Inadequate simulation: One of the techniques based on extendable computer simulator making it too shorter time than it takes to develop the complete design.

(b). Deficient design representation: Machine process able structured design languages are more suitable to understand and maintain.

(c). Unstructured ness: Structured ness refers to design philosophy requiring adherence to a set of rules of enforced standards which embody techniques as top-down design, program modularization or independence and so forth.

(d). Selection of un-standardized language: Standardization implies, among other requirements that rigid configuration controls be kept on compilers, support and documentation. An un-standardized language can be the source of coding errors.

10.3.2. Coding Errors

(a) Topographical errors: Errors formed due to incorrectly writing down or copying a statement in the source language.

(b) Data Structure defects: Program incompatible with the data structures specification.

(c) Algorithmic approximation: Approximations may be insufficiently accurate over the required ranges of the variables.

(d) Misinterpretation of language constructions: Thinking certain program language to be correct, programmer uses it, but the compiler interprets them differently.

(e) Missing Incorrect logic: Assuming that the specification to be correct, programmer makes an error by omitting a required test for a condition.

(f) Undocumented Assumptions: Programmer makes an assumption in design interpretation which later results in two or more interpretations.

10.3.3. Clerical Errors

(a) Manual error

(b) Mental error

(c) Procedural error

(d) Other clerical errors

10.3.4. Debugging errors

(a) Inadequate use of Debugging Tools

(b) Insufficient or inappropriate selection of test data

(c) Misinterpretation of Debugging results

(d) Misinterpretation of Error source

(e) Negligence

(f) Other Debugging errors

10.3.5. Testing errors

(a) Inadequate test cases or test data

(b) Misinterpretation of test results

(c) Misinterpretation of program specification

(d) Negligence

(e) Other testing errors

10.4. Failure Classification

(a) ***Transient***: Occurs only with certain inputs.

(b) ***Permanent***: Occurs with all inputs.

(c) ***Recoverable***: System can recover without operator intervention.

(d) ***Unrecoverable***: Operator intervention needed to recover from failure.

(e) ***Non-corrupting***: Failure does not corrupt systems.

(f) ***Corrupting***: Failure corrupts system state or data.

10.5. Data Collection

A proper collection and analysis of software failure data lies at the heart of a practical evaluation of the quality of software based systems. This section provides insight into the process of collection of software failure data. Measurements typically involve recording the times between successive failures of the software when it is executing in a simulated or operational environment.

Measurements can be taken in terms of:

- Execution Time - the actual processing time for the execution of the program
- Calendar Time - the time in familiar terms of seconds, minutes, and hours.
- Clock Time - the time a computer is running while executing the program. Other programs may be executing on the same machine at the same time.

10.5.1. Data collection procedure

Step1: Establish the objective

This is the first step in planning to collect data is to determine the objectives of the data and what data items will be collected. Data collection does involve cost, so each item should be examined to see if the need is worth the cost. This should be done in the context of the planned application or the applications of the software reliability engineering. If the item is questionable, consider alternative such as approximating the item or collecting t at a lower frequency. Look for possibilities of collecting data items that can serve multiple purposes. If this careful examination is not performed,

the necessary burden in effort and cost on the project can result in the degradation of all data or even the abandonment of the effort.

Step 2: Plan the data collection process

It is recommended that all parties (designers, coders, testers, users, and key management) participate in the planning effort. The data collectors must be motivated if quality data is to collected. Present the goals of the data collection effort. Relate to it direct personal benefit. This will insure that all parties understand what is being done and the impact it will have on their respective organizations.

It is suggested that a first draft data collection plan be presented as a starting point. The plan should include topics such as:

- What data items will be gathered?
- Who will gather the data?
- How often will that data be reported?
- Formats of data reporting (e.g. electronic spreadsheet, and paper forms)
- How is the data to be stored and processed?
- How will the collection process be monitored to ensure integrity of the data?

Solicit identification of problems with the plan and desired improvements. Elicit the participation of the data collectors in the solution of any problems. It will provide them an opportunity to provide new ideas and insight into the development of process. Support will be gained by having the parties that will be affected as active participants.

Recording procedures should be carefully considered to make them as simple as possible. Solicitation of data from project members can reduce effort and make collection more reliable.

For the failure count methods, the data collection interval should be selected to correspond to the normal reporting interval of the project from which data are being collected (e. g. week, month) or an integral multiple thereof. This will facilitate obtaining data on the level of effort devoted to the software under test (person-hours and computer hours) which must be correlated with the reliability data.

Step 3: Apply tools

Availability of tools identified in the collection process must be considered. If the tools are not commercially available then time needs to be planned for their development. Furthermore, the amount of automatic data collection must be considered. To minimize the impact on the project's schedule, automated tools should be considered whenever possible.

When decision are being made to automate the data collection process for either of the two types of data one needs to weigh certain factors. These include:

- Availability of the tool. Can it be purchased or must 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 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 computer time) will be needed to keep the data collection process going?

Once the tool has been developed and implemented, one needs to consider ways of ensuring the right data being gathered. Flexibility also should be designed into the tools, as the data collection requirements may change. Finally, one needs to a make some type of assessment of not only what the tool saved in time and resources but also what data collection process gained. Records could be kept of the number of faults detected after the release of the software. This could be compared with reliability estimates of similar projects that did not employ this methodology. Estimates of reduced maintenance and fault correction time could be made based on upon the estimated current failure rate.

For the tool itself, one could estimate the amount of time and effort that would be expended if the data had been collected manually. These statistics could then yield

cost estimates which would be compared with the procurement and implementation costs of the automated tool. If the cost of the automated tool is significantly higher, one certainly would question the wisdom of developing the tool. However, even if the costs come out higher, consideration must be given to future use of the tool. Once the tool has been developed it may be easily adapted over many software development efforts and could yield significant savings.

Step 4: Provide training.

Once the tools and plans are in place, training of all concerned parties is important. The data collectors need to understand the purpose of the measurements and know explicitly what data are to be gathered.

Step 5: Perform trial run.

A trial run of the data plan should be made to resolve any problems or misconceptions about the plan. This can save vast amount of time and effort when the "real thing" occurs.

Step 6: Implement the plan.

Data must be collected and reviewed promptly. If this is not done, quality will suffer. Generate reports to show project members; they can often spot unlikely results and thus identify problems. Problems should be resolved quickly before the information required to resolve them disappears.

Step 7: Monitor data collection.

Monitor the process as it proceeds to insure the objectives are met and the program is meeting its established reliability goals.

Step 8: Use the data.

Don't wait to the end after the software has been released to the users to make your reliability assessments. Estimating software reliability at regular, frequent intervals will maximize visibility into the development effort, permitting managerial decisions to be made on a regular basis.

Step 9: Provide feedback.

This should be done as early as possible during the data collection. It is especially important to do so at the end. Those who were involved want to hear what impact their efforts had. If no feedback is given, you'll find yourself facing the problem alluded to in the beginning of this section. Namely, the parties will resist further future efforts because they see no purpose.

10.6. Failure Count Data vs. Execution Time Data

It is generally accepted that execution (CPU) time is superior to calendar time for software reliability measurement and modeling. If execution time is not readily available, approximations such as clock time, weighted clock time, or units that are naturals to the applications, such as transactions, may be used.

The following paragraphs address failure count and execution time data collection to support the recommended models identified.

10.6.1. Failure-Count Data

Since the recommended models employ the number of failures detected per unit of time, these data are usually readily available. Most organizations have some type of configuration management process in place. As part of this process, a procedure for reporting failures and approving changes to the software is in place. The software problem reporting mechanism may be either manual or automatic. In addition, the problem reports may be stored within a computer data base system or a manual filing system. The key is that the data can be easily extracted.

Make sure that the problems are really software problems - some organizations use problem reporting for any type of anomaly and the time recorded on a problem report may not be the time at which the failure was experienced, it may be the time in which the port was filled out

Another pitfall to avoid when using problem reporting data involves forming the time intervals. Remember, the purpose is to model the number of failures detected per unit of time within a specified environment. These units should therefore be consistent in duration, manpower, and testing intensity.

Usually the information to check this is not available. All one has is data on the number of failures detected in one period or another. However, there may have been

twice as many testing personnel in one period than the other. The only way to find out this information is to seek it out. This may involve talking with the testers or even reviewing old time sheets covering the period of interest. Generally, the longer the period of time in which the fault counts are formed the more smoothing occurs. Variations within short intervals of time will be averaged out over the longer time units.

Data may be gathered at any point within the development cycle beginning with the system test phase. Overall measurement objectives will help you determine the rate (failures reported per week, per month, or per quarter), at which data is collected. It is suggested that you start out using the number of failures reported over the shortest unit of time consistent with your objectives. If good fits are not achieved, then combine intervals to the next level. For example: days to weeks, or weeks to quarters. The smoothing effect mentioned in the previous paragraph may help in the modeling process.

10.6.2. Execution Time Data

This data may be collected directly or indirectly. Also, it is best to collect, when feasible, the actual execution time of a program rather than the amount of wall clock time or system active time expended. This is the actual amount of time spent by the processor in executing the instructions. Execution time gives a truer picture of the stress placed on the software. You could have large amounts of time expended on the clock but very little computations may have to be done during this period. This yields small execution times. This would tend to give overly optimistic views of the reliability of the software. Modeling using execution time data tends to give superior results than simple elapsed wall clock time or system active time. However, the data may be difficult to collect since a monitor of the actual operating system is involved. Another source for obtaining this data is to adjust the wall clock time by a factor that represents the average computer utilization per unit of wall clock time.

If the time between failures (wall clock or execution time) is unavailable and only grouped data (number of failures occurring per unit of time) is available, the time-between-failures can still be obtained. One way is to randomly allocate the failures over the length of the time interval. Randomization will not cause errors in estimation for some of the models by more than 15 percent. A second way is the

easiest to implement. Simply allocate the failures uniformly over the interval length. For example, suppose the interval is three hours in duration and three failures occurred during this period. We could then treat the time-between-failures to be each one hour in length.

Two additional considerations are: (1) adjusting the failure times to reflect an evolving program and (2) handling multiple versions of the software. In the first situation, the failure intensity may be underestimated in the early stages of the program's development yielding overly optimistic views of the reliability. For the second consideration, there are multiple versions of the code being executed at different locations.

10.6.3. Transformations between the Two Types of Input

Programs may have the capability to estimate model parameters from either failure-count or time-between-failures data, as maximum likelihood estimation can be applied' to both. However, if a program accommodates only one type of data it is easy to transform to the other type.

If the expected input is failure-count data, it may be obtained by transforming time-between-failures data to cumulative time data and then simply counting the cumulative times that occur within a specified time period.

The expected input is time-between-failures data, convert the failure-count data by randomly selecting a number of cumulative failures times in the period equal to the count and then finding the time differences between them.

10.7. Software Reliability Engineering

Software Reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. Software reliability is an attribute and key factor in software quality. It is also a system dependability concept. Software Reliability Engineering (SRE) is defined as the quantitative study of the operational behavior of software-based systems with respect to user requirements concerning reliability. SRE employs proven best practice to ensure that product reliability meets user needs, to speed products to market faster, reduce product cost, improve customer satisfaction and increase tester and developer productivity.

Essential Components of SRE:

1. Establish reliability goals.
2. Develop operational profile.
3. Plan and execute tests.
4. Use test results to drive decisions.

These components are sequential but they are integrated within the software development process.

10.7.1. What It Is and Why It Works

Let's look in a little more depth now at just what SRE is. SRE is a practice for quantitatively planning and guiding software development and test, with emphasis on reliability and availability. It is a practice that is backed with science and technology (Musa, Iannino, and Okumoto (1987)). But we will describe how it works in business-oriented terms.

SRE works by quantitatively characterizing and applying two things about the product: the expected relative use of its functions and its required major quality characteristics. The major quality characteristics are reliability, availability, delivery date, and life-cycle cost. In applying SRE, you can vary the relative emphasis you place on these factors.

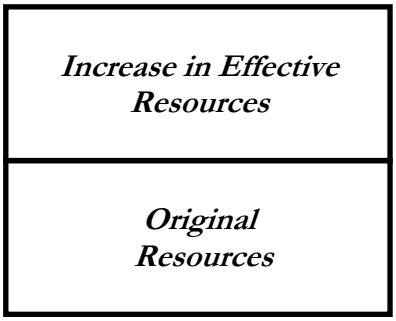


Figure 10.2: Increased resource pool resulting from increased development efficiency

When you have characterized use, you can substantially increase development efficiency by focusing resources on functions in proportion to use and criticality. You also maximize test effectiveness by making test highly representative of use in the field. Increased efficiency increases the effective resource pool available to add customer value, as shown in Figure 10.2.

When you have determined the precise balance of major quality characteristics that meets user needs, you can spend your increased resource pool to carefully match them. You choose software reliability strategies to meet the objectives, based on data collected from previous projects. You also track reliability in system test against its objective to adjust your test process and to determine when test may be terminated. The result is greater efficiency in converting resources to customer value, as shown in Figure 10.2. We have set delivery times and budgeted software costs for software-based systems for some time. It is only relatively recently that SRE, the technology for setting and tracking reliability and availability objectives for software, has developed (Musa, Iannino, and Okumoto 1987).

10.7.2. A Proven, Standard, Widespread Best Practice

Software reliability engineering is a proven, standard, widespread best practice. As one example of the proven benefit of SRE, AT&T applied SRE to two different releases of a switching system, International Definity PBX. Customer-reported problems decreased by a factor of 10, the system test interval decreased by a factor of 2, and total development time decreased 30%. No serious service outages occurred in 2 years of deployment of thousands of systems in the field (Lyu 1996).

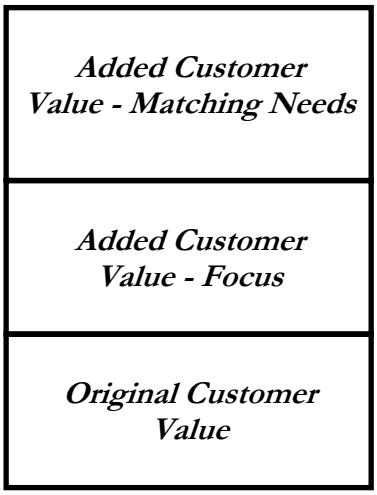


Figure 10.3: Increased customer value resulting from increased resource pool and better match to major quality characteristics needed by users.

SRE has been an AT&T Best Current Practice since May 1991 (Lyu 1996). To become a Best Current Practice, a practice must have substantial application (usually at least 8 to 10 projects) and this application must show a strong, documented benefit-to-cost ratio. For SRE, this ratio was 12 or higher for all projects. The

practice undergoes a probing review by two boards, at third and fourth levels of management. More than 70 project managers or their representatives reviewed the SRE proposal. There were more than 100 questions and issues requiring resolution, a process that took several months. In 1991, SRE was one of five practices that were approved, out of 30 that were proposed.

SRE is widely applicable. From a technical viewpoint, you can apply SRE to any software-based product, starting at the beginning of any release cycle. From an economic viewpoint, you can apply SRE to any software-based product also, except for very small components, perhaps those involving a total effort of less than 2 staff months. However, if a small component such as this is used for several projects, then it probably will be feasible to use SRE. If not, it still may be worthwhile to implement SRE in abbreviated form.

Table 10.1: Operating cost of SRE

Project size (staff years)	Percent of project cost
5	3
10	2
20	1.5
100	0.4
500	0.1

SRE is independent of development technology and platform. It requires no changes in architecture, design, or code, but it may suggest changes that would be useful. It can be deployed in one step or in stages. SRE is very customer-oriented: it involves frequent direct close interaction with customers. This enhances a supplier's image and improves customer satisfaction, greatly reducing the risk of angry customers. Developers who have applied SRE have described it with adjectives such as "unique, powerful, thorough, methodical, and focused." It is highly correlated with attaining Levels 4 and 5 of the Software Engineering Institute Capability Maturity Model.

Despite the word "software," software reliability engineering deals with the entire product, although it focuses on the software part. It takes a full-life-cycle, proactive view, as it is dependent on activities throughout the life cycle. It involves system engineers, system architects, developers, users (or their representatives, such

as field support engineers and marketing personnel), and managers in a collaborative relationship.

The cost of implementing SRE is small. There is an investment cost of not more than 3 equivalent staff days per person in an organization, which includes a 2-day course for everyone and planning with a much smaller number. The operating cost over the project life cycle typically varies from 0.1 to 3 percent of total project cost, as shown in Table 10.1. The largest cost component is the cost of developing the operational profile.

The schedule impact of SRE is minimal. Most SRE activities involve only a small effort that can parallel other software development work. The only significant critical path activity is 2 days of training.

SRE differs from other approaches by being primarily quantitative. In applying SRE, you add and integrate it with other good processes and practices; you do not replace them. With SRE you control the development process, it doesn't control you. The development process is not externally imposed. You use quantitative information to choose the most cost-effective software reliability strategies for your situation.

Before we proceed further, let's define some of the terms we will be using. Reliability is the probability that a system or a capability of a system functions without failure for a specified period in a specified environment. The period may be specified in natural or time units.

The concept of natural units is relatively new to reliability, and it appears to have originated in the software sphere. A natural unit is a unit other than time that is related to the amount of processing performed by a software-based product, such as pages of output, transactions, telephone calls, jobs, semiconductor wafers, queries, or application program interface calls. *Availability* is the average (over time) probability that a system or a capability of a system is currently functional in a specified environment. If you are given an average down time per failure, availability implies a certain reliability. *Failure intensity*, used particularly in the field of software reliability engineering, is simply the number of failures per natural or time unit. It is an alternative way of expressing reliability.

10.8. Software Reliability Measurements

Measurements of reliability includes two types of activities

i. Reliability estimation

ii. Reliability prediction

10.8.1. Software reliability estimation

This activity determines current software reliability by applying statistical inference techniques to failure data obtained during system test or during system operation. This is a measure regarding the achieved reliability from the past until the current point.

10.8.2. Software reliability prediction

This activity determines future software reliability based upon available software metrics and measures. The quality of software, and in particular its reliability, can be measured in terms of metrics of failure intensity or mean time between failures (MTBF). Mean time between failures can be approximated by the inverse of the failure intensity. When there is no repair it may be possible to describe the reliability of the software-based system using constant failure intensity, λ, and an exponential relationship.

$$R(t) = e^{-\lambda\tau}$$

where,

R(t)- Reliability of the system.

τ- Duration of the mission

λ- Failure rate

$$\text{MTTF} = \int_0^\infty e^{-\lambda\tau} d\tau$$

$$= 1/\lambda \text{ (Constant failure rate)}$$

10.9. Type of Tests in SRE

There are two types of software reliability engineering test, reliability growth test and certification test. These types are not related to phases of test such as unit test, sub system test, system test or beta test, but rather to the objectives of the test.

10.9.1. Reliability growth test

This test is used to estimate and track reliability. The main objective of reliability growth test is to find and remove faults. Reliability growth test is used for the system test phase of the software developed in own organization. Testers and development managers apply the reliability information to guide development and release. To obtain good (with moderate ranges of uncertainty) estimates of failure intensity, one needs a minimum number of failures in sample, often 10 to 20. Reliability growth test includes feature, load and regression test.

Feature test is a test in which operations are executed separately, with interactions and effects of the field environment minimized by reinitializing the system between the operations. The idea is to verify all features of the software.

Regression test is the execution of some (usually randomly selected) or all feature tests after each system build that has a significant change. One should include all critical operations in the regression test suite.

Load test involves executing operations simultaneously, at the same rates and with the same other environmental conditions as those that will occur in the field. Thus the same interactions and impact of environmental conditions will occur as can be expected in the field. Acceptance test and performance test are types of load test. Load test typically involves competition for system resources with the queuing and timing problems that can result. Also, there is frequently a degradation of data with time. The foregoing factors thus can uncover potential field failures resulting from interaction that would not be stimulated by feature and regression test.

10.9.2. Certification test

This test does not involve debugging. There is no attempt to resolve failures identified by determining the faults that are causing them and removing the faults. The system must be stable. No changes can be occurring, either due to new features or fault removal. With certification test one makes a binary decision to accept the software or reject the software and return to its supplier for rework. In certification test; one requires a much smaller sample of failures. Certification testing is dealt in chapter 7. In certification test only load test (not feature or regression test) is generally done.

10.10. Software Reliability Engineered Testing

The advance in technology, the emergence of a wide variety of software applications, and the increase in the use of computer systems have led to an increase in demand for higher standards in software quality and reliability. At the same time, the software market competitiveness is increasing very rapidly, so software products will not succeed in the market unless they are produced with high quality standards. Indeed, high quality in computer systems leads to increased productivity and permanently reduced costs by emphasizing fault prevention.

Additionally, there is a diverse use of computer-based systems ranging from commercial applications and automobile controls to medical devices, aircrafts, space and nuclear reactor controls, and software failures or incorrect software requirements can have disastrous consequences ranging from the loss of financial assets and customer dissatisfaction to the harming or the loss of human lives. Therefore, software engineering provides a ”tool kit" needed for successful construction of high quality software, so that useful, reliable and safe software can be released on time within a budget. Even though some systems are less market-driven than others it is important to balance reliability, time of delivery and cost, and one of the most effective ways of achieving that goals is through engineering of testing using quantitative planning and tracking.

Software reliability engineered testing (SRET) is a technique introduced by Musa which combines quantitative reliability objectives and operational profiles, so developers can have a more realistic guide when performing testing. In this way, it is possible to track the reliability that is actually being achieved thought the software life cycle.

Software reliability is defined as “the probability of execution without failure for some specific interval, called “mission time”. It is observed that this definition is closely related to the definition of hardware reliability with the difference that the failure mechanism may be different. The reason for that compatibility is that in a system we have software and hardware components; therefore, software system can be referred to as a software-based system.

When applying SRET to software systems, it must be done over the whole software life cycle, with particular emphasis on the testing phase. Also, if it is a legacy system, it must be applied to all the releases as well. Usually, testers are the people most involved in the process, but better results can be obtained by involving system engineers, architects and uses as well.

SRET consists of seven steps; the first two steps consist of decision-making and are the foundation for the subsequent five core steps. Those steps are the following : (1) determine which associated systems require separate testing, (2) decide which type of SRET is need for each system to be tested, (3) define "necessary reliability", (4) develop operational profiles, (5) prepare for testing, (6) execute tests, and (7) interpret failure data. The five core steps and the life cycle phase in which each step typically occurs are shown in Figure 10.4.

10.10.1. Definitions

Before continuing with the description of the steps of SRET, we consider of importance to define the different types of testing referred further in the paper. When using SRET, we can consider two types of testing: development testing and certification testing. In developing testing, faults are found and removed. It includes feature, load and regression testing. In feature testing, test runs are executed undependably of each other, while in load testing a large number of test cases are executed in the context of operation mode. In regression testing, feature test runs

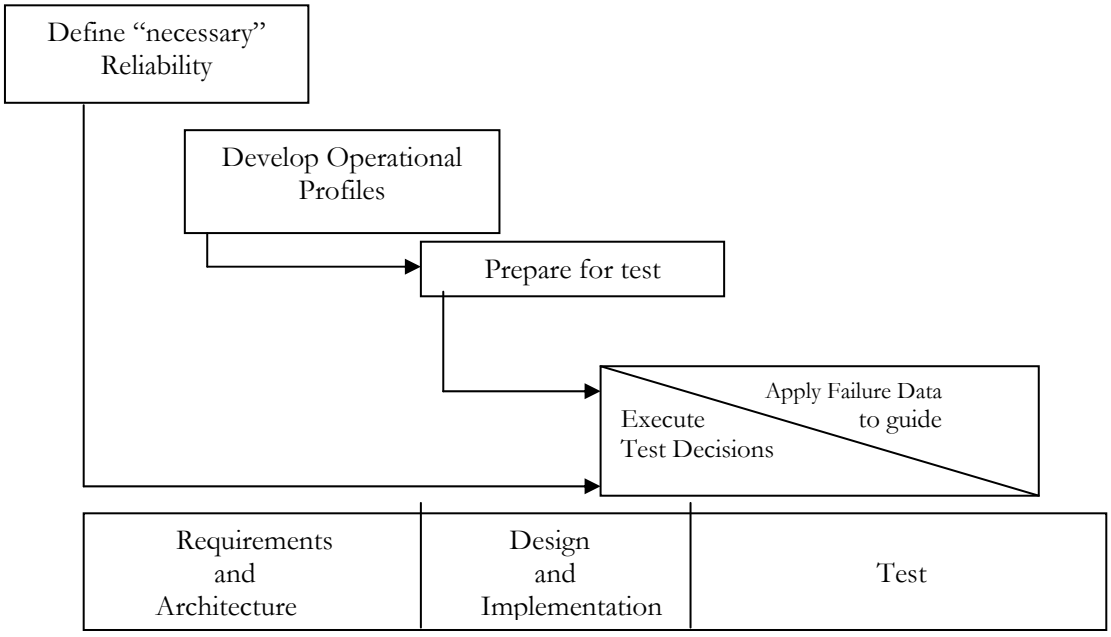


Figure 10.4: Software Reliability Engineering process diagram

are repeated after each build to verify whether changes to the system have introduced faults that cause failure. In certification testing, the software is either rejected or accepted and does not remove any faults.

10.10.2. SRET Steps

1. ***Determining which associated systems require separated testing:*** In this step, system variations as well as major components of unknown reliability may need to be tested in addition to the entire system. In addition, small components that may need to be reused extensible and software system that interact strongly with the system may need to be tested in a way they are functioning together.
2. ***Decide the type of SRET need for each system to be tested:*** Decide which type of testing the system or related system may need. Development testing can only be applied to the system being implemented or at least coded in part.
3. ***Define the "necessary" reliability:*** This step consists of the following steps:

   a) ***Determining operational modes:*** An operational mode is defined as a distinct pattern of system use and/or environment that needs separate testing because it is likely to stimulate different failures, which can also be established to provide accelerated testing of rarely occurring but critical operations. Some of the factors that may yield operational modes are: day of the week or time of the day, time of the year, traffic levels, user profile, user experience, system maturity, reduced system capability, and rarely critical events. However, the selection of operational profiles is based in engineering judgment, and a lot of attention has to be paid to the trade-offs of selecting a big or small amount of operational profiles, i.e., increase in the number of operational modes can provide better and more realistic testing, but it cal also increase cost and effort.

   b) ***Define failure in terms of severity classes:*** A *failure* is defined as the departure of a program's behavior during execution form the user requirements and a *fault* is the defect in the program that triggers the failure when executed. A severity class is a set of failures that affect users to the same degree. It is usually related to the criticality of the operation that fails and the common classification criteria may include the degree of impact on cost, service and human life.

   c) ***Set failure intensity objectives for the developed software:*** This step consists of the following steps:

      i) Establish the system failure objectives, which can be derived from and analysis of specific user needs and the capabilities of competing systems;

ii) Determine and sum the failure intensities of the acquired software and hardware components;

iii) Subtract the total acquired failure intensities from the system failure intensity objectives in clock hours;

iv) Convert the results into failure intensity objectives for the developed software per unit of execution time.

d) ***Engineer reliability strategies:*** There are three main reliability strategies: *fault prevention, fault removal and fault tolerance*. Fault prevention is about trying to reduce faults by performing requirements and design reviews. Fault removal tries to detect and eliminate faults from the system using code inspection and development testing. Fault tolerance tries to minimize the number of failures in the system by detecting and accounting for deviations in the programs execution that may lead to failures.

4. ***Develop operational profiles:*** There are two types of operational profiles: overall mode, which is used to select the test cases to prepare and operational model, which is used to select operations for execution when a specific operational mode is tested. When developing operational profiles the following steps are used [8].

a) Identify the initiators of operations: First, it is necessary to identify the expected customer types on the basis of information such as system business case and marketing data for related systems and for set of users that tend to use the system in the same way.

b) List the operations each initiator produces: The system requirements, and sources such as work flow diagrams; user manuals, prototypes and pervious versions of the system can be used to extract the list of operations.

c) Determine the occurrence rate per clock hour of the operations: This information can be obtained from already existing field data from previous versions or similar systems; otherwise it can be collected. Also, if the operations are event driven, simulated environments can be created to determine the event frequency.

d) Determine the occurrence probability: In this step, divide the occurrence rated for each operation by the total operation occurrence rates.

5. Prepare for testing: A *run* is defined as a specific instance of an operation and is characterized by that operation and a complete set of values of its input variables. An input variable is the one that exists external to the run, but influences it. Input variables can be direct or indirect. Direct input variables control

processing in a known and designed way, while indirect input variables influence the processing, but in an unpredictable way. Runs differ from test cases because in a test case only the direct input variables are provided, and it becomes a run when the indirect input variables are provides too. When preparing for testing the following steps are applied:

a) Specify test cases: The number of test cases has to be specified in a way that it is cost-effective. When selecting the test cases the following steps are used:

i) Select the operations depending on their occurrence probabilities. It can be determined using the operational profile with proper modifications acceding to the critical operations.

ii) Complete the selection of test cases by choosing the level for all direct input variables, which is the value or range of values on input variables for which failure behavior is expected to be the same because of processing similarities. After, selecting all the levels for each direct input variable, randomly choose a lever for each direct input variable.

b) Define test procedures: A test procedure is the specification of the set of runs and environment associated with an operational mode.

c) Prepare automated tools: For SRET, it is not a requirement to have automated tools, but failure-identifications tools usually make the process faster and more efficient.

6. Execute tests: First, start with feature testing a follow with load testing. Use regression testing after each build involves significant change. This step, includes identifying failures, determining when they occurred and establishing the severity of their impact. Also, it is important to determine the rime of failure occurrence or number of failures per time period in execution time.

7. ***Interpret failure data:*** The failure data is interpreted at two different levels, depending on the type of testing being used in a system component.

   a) ***Development testing:*** Finding trends using estimation and plotting the failure intensity over all the severity classes and across all the operational modes against calendar time is a typical activity in this step. Comparing the overall failure intensity objective can aid in the identification of schedules or reliabilities that are at risk , so the proper corrective measures can be taken.

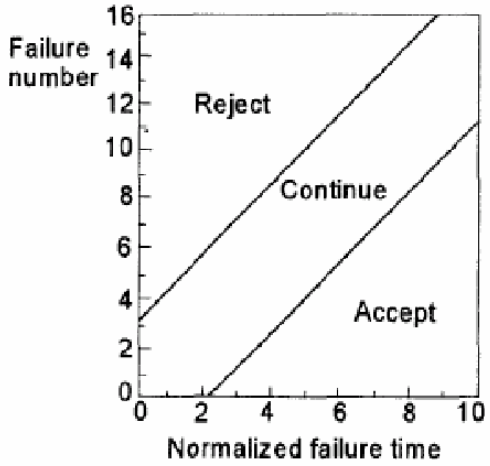


Figure 10.5: Failure Interpretation for Certification Testing

b) Certification testing: Figure 10.5 shows the typical graphical representation for interpreting failure when performing certification testing. To normalize the failure times, multiply it by the appropriate failure intensity objective. Reliability charts can be built for different levels of consumer risk and supplier risk.

10.11. SRE Process and Fone Follower Example

Let's now take a look at the SRE process. There are six principal activities, as shown in Figure 10.6. We show the software development process below and in parallel with the SRE process, so you can relate the activities of one to those of the other. Both processes follow spiral models, but we don't show the feedback paths for simplicity. In the field, we collect certain data and use it to improve the SRE process for succeeding releases.

The List Associated Systems, Implement Operational Profiles, Define "Just Right" Reliability, and Prepare for Test activities all start during the Requirements and Architecture phases of the software development process. They all extend to varying degrees into the Design and Implementation phase, as they can be affected by it. The Execute Test and Guide Test activities coincide with the Test phase.

We will illustrate the SRE process with Fone Follower, an example adapted from an actual project at AT&T. We have changed the name and certain details to keep the explanation simple and protect proprietary data. Subscribers to Fone Follower call and enter, as a function of time, the phone numbers to which they want to forward their calls. Fone Follower forwards a subscriber's incoming calls (voice or fax) from the network according to the program the subscriber entered. Incomplete voice calls go to the subscriber's pager (if the subscriber has one) and then, if unanswered, to voice mail. If the subscriber does not have a pager, incomplete voice

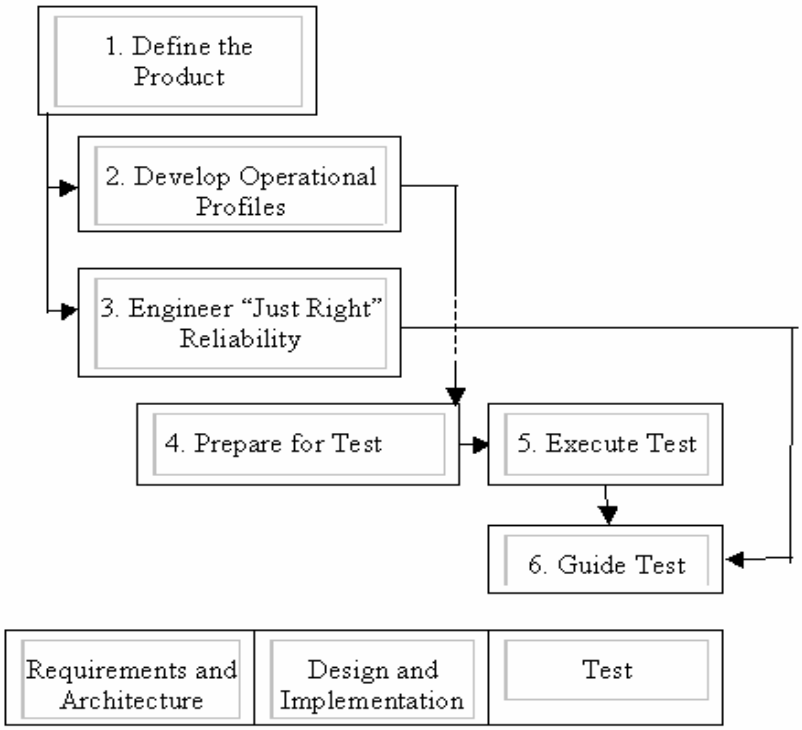


Figure 10.6: The Core Application Steps of SRET and the Corresponding Development Life Cycle Phase

calls go directly to voice mail.

10.11.1. Define the Product

The first activity is to define the product. You must establish who the supplier is and who the customers and users are, which can be a nontrivial enterprise in these days

Table 10.2: Fone Follower Operational Profile

Operation	Occurrence Probability
Proc. voice call, no pager, ans.	0.21
Proc. voice call, pager, ans.	0.19
Proc. fax call	0.17
Proc. voice call, pager, ans. on page	0.13
Proc. voice call, no pager, no ans.	0.10
Proc. voice call, pager, no ans. on page	0.10
Enter forwardees	0.09
Audit sect. - phone number data base	0.009
Add subscriber	0.0005
Delete subscriber	0.0005
Recover from hardware failure	0.000001
Total	1

of outsourcing and complex inter- and intra company relationships. Then you list all the systems associated with the product that for various reasons must be tested independently.

10.11.2. Implement Operational Profiles

This section deals with quantifying how software is used. To fully understand it, we need to first consider what operations and operational profiles are.

An *operation* is a major system logical task, which returns control to the system when complete. Some illustrations from Fone Follower are Phone number entry, Process fax call, and Audit a section of the phone number data base. An *operational profile* is a complete set of operations with their probabilities of occurrence. Table 10.2 shows an illustration of an operational profile from Fone Follower.

There are five principal steps in developing an operational profile:

1. Identify the operation initiators
2. List the operations invoked by each initiator
3. Review the operations list to ensure that the operations have certain desirable characteristics and form a set that is complete with high probability
4. Determine the occurrence rates
5. Determine the occurrence probabilities by dividing the occurrence rates by the total occurrence rate

There are three principal kinds of initiators: user types, external systems, and the system itself. You can determine user types by considering customer types. For

Fone Follower, one of the user types is subscribers and the principal external system is the telephone network. Among other operations, subscribers initiate Phone number entry and the telephone network initiates Process fax call. Fone Follower itself initiates Audit a section of the phone number data base.

When implementing SRE for the first time, some software practitioners are initially concerned about possible difficulties in determining occurrence rates. Experience indicates that this is usually not a difficult problem. Software practitioners are often not aware of all the use data that exists, as it is typically in the business side of the house. Occurrence rate data is often available or can be derived from a previous release or similar system. New products are not usually approved for development unless a business case study has been made, and this must typically estimate occurrence rates for the use of various functions to demonstrate profitability. One can collect data from the field, and if all else fails, one can usually make reasonable estimates of expected occurrence rates. In any case, even if there are errors in estimating occurrence rates, the advantage of having an operational profile far outweighs not having one at all.

Once you have developed the operational profile, you can employ it, along with criticality information, to:

1. Review the functionality to be implemented for operations that are not likely to be worth their cost and remove them or handle them in other ways (Reduced Operation Software or ROS)
2. Suggest operations where looking for opportunities for reuse will be most cost-effective
3. Plan a more competitive release strategy using operational development. With operational development, development proceeds operation by operation, ordered by the operational profile. This makes it possible to deliver the most used, most critical capabilities to customers earlier than scheduled because the less used, less critical capabilities are delivered later.
4. Allocate development resources among operations for system engineering, architectural design, requirements reviews, and design to cut schedules and costs.
5. Allocate development resources among modules for code, code reviews, and unit test to cut schedules and costs

6. Allocate new test cases of a release among the new operations of the base product and its variations
7. Allocate test time

10.11.3. Define "Just Right" Reliability

To define the "just right" level of reliability for a product, you must first define what "failure" means for the product. We will define a *failure* as any departure of system behavior in execution from user needs. You have to interpret exactly what this means for your product. The definition must be consistent over the life of the product, and you should clarify it with examples. A failure is not the same thing as a fault; a *fault* is a defect in system implementation that causes the failure when executed. Beware, as there are many situations where the two have been confused in the literature.

The second step in defining the "just right" level of reliability is to choose a common measure for all failure intensities, either failures per some natural unit or failures per hour.

Then you set the total system failure intensity objective (FIO) for each associated system. To determine an objective, you should analyze the needs and expectations of users.

For each system you are developing, you must compute a developed software FIO. You do this by subtracting the total of the expected failure intensities of all hardware and acquired software components from the system FIOs. You will use the developed software FIOs to track the reliability growth during system test of all the systems you are developing with the failure intensity to failure intensity objective (FI/FIO) ratios.

You will also apply the developed software FIOs in choosing the mix of software reliability strategies that meet these and the schedule and product cost objectives with the lowest development cost. These include strategies that are simply selected or not (requirements reviews, design reviews, and code reviews) and strategies that are selected and controlled (amount of system test, amount of fault tolerance). SRE provides guidelines and some quantitative information for the

determination of this mix. However, projects can improve the process by collecting information that is particular to their environment.

10.11.4. Prepare For Test

The Prepare for Test activity uses the operational profiles you have developed to prepare test cases and test procedures. You allocate test cases in accordance with the operational profile. For example, for the Fone Follower base product there were 500 test cases to allocate. The Process fax call operation received 17 percent of them, or 85.

After you assign test cases to operations, you specify the test cases within the operations by selecting from all the possible intraoperation choices with equal probability. The selections are usually among different sets of values of input variables associated with the operations, sets that cause different processing to occur. These sets are called *equivalence classes*. For example, one of the input variables for the Process fax call operation was the Forwarded (number to which the call was forwarded) and one of the equivalence classes of this input variable was Local calling area. You then select a specific value within the equivalence class so that you define a specific test case.

The test procedure is the controller that invokes test cases during execution. It uses the operational profile, modified to account for critical operations and for reused operations from previous releases.

10.11.5. Execute Test

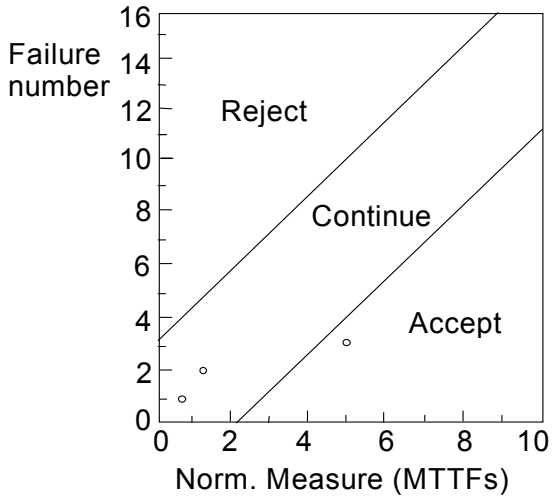
In the Execute Test activity, you will first allocate test time among the associated systems and types of test (feature, load, and regression).

Invoke feature tests first. *Feature tests* execute all the new test cases of a release independently of each other, with interactions and effects of the field environment minimized (sometimes by reinitializing the system). Follow these by *load tests*, which execute test cases simultaneously, with full interactions and all the effects of the field environment. Here you invoke the test cases at random times, choosing operations randomly in accord with the operational profile. Invoke a regression test after each build involving significant change. A *regression test* executes some or all feature tests; it is designed to reveal failures caused by faults introduced by program changes.

Identify failures, along with when they occur. The "when" can be with respect to natural units or time. This information will be used in Guide Test.

10.11.6. Guide Test

The last activity involves guiding the product's system test phase and release. For software that you develop, track reliability growth as you attempt to remove faults. Then we certify the super systems, which simply involves accepting or rejecting the software in question. We also use certification test for any software that we expect



FAIL. NO.	MCALLS AT FAILURE	MTTFs
1	0.1875	0.75
2	0.3125	1.25
3	1.25	5

Failure intensity objective:
4 failures / Mcalls

Figure 10.7: Reliability Demonstration Chart Applied to Fone Follower

customers will acceptance test.

For certification test you first normalize failure data by multiplying by the failure intensity objective. The unit "Mcalls" is millions of calls. Plot each new failure as it occurs on a reliability demonstration chart as shown in Figure 10.7. Note that the first two failures fall in the Continue region. This means that there is not enough data to reach an accept or reject decision. The third failure falls in the Accept region, which indicates that you can accept the software, subject to the levels of risk associated with the chart you are using. If these levels of risk are unacceptable, you construct another chart with the levels you desire (Musa 2004) and re-plot the data.

To track reliability growth, input failure data that you collect in Execute Test to a reliability estimation program such as CASRE, normalize the data by multiplying by the failure intensity objective in the same units. Execute this program periodically and plot the FI/FIO ratio as shown in Figure 10.8 for Fone Follower. If you observe a significant upward trend in this ratio, you should determine and correct the causes. The most common causes are system evolution, which may indicate poor change control, and changes in test selection probability with time, which may indicate a poor test process.

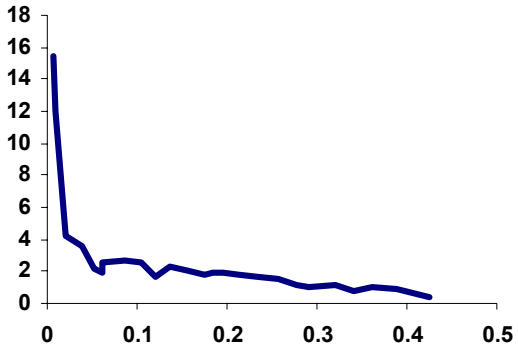


Figure 10.8: Plot of FI/FIO Ratio for Fone Follower

If you find you are close to your scheduled test completion date but have an FI/FIO ratio substantially greater than 0.5, you have three feasible options: defer some features or operations, rebalance your major quality characteristic objectives, or increase work hours for your organization. When the FI/FIO ratio reaches 0.5, you should consider release as long as essential documentation is complete and you have resolved outstanding high severity failures (you have removed the faults causing them).

Developers sometimes worry that systems with ultrareliable FIOs might require impractically long hours of test to certify the FIOs specified. But there are many ameliorating circumstances that make the problem more tractable than that for ultra-reliable hardware (Musa 2004). First, in most cases only a few critical operations, not the entire system, must be ultra-reliable. Second, software reliability relates to the execution time of the software, not the clock time for which the system is operating as does hardware. Since the critical operations often occur only rarely, the execution time of the critical operations is frequently a small fraction of the clock time. Thus

the FIO for the entire system need not be ultra-reliable. Finally, since processing capacity is cheap and rapidly becoming cheaper, it is quite feasible to test at a rate that is hundreds of times real time by using parallel processors. Thus testing of ultra-reliable software can be manageable.

10.11.7. Collect Field Data

The SRE process is not complete when you ship a product. We collect certain field data to use in succeeding releases and in other products. In many cases, we can collect the data easily and inexpensively by building recording and reporting routines into the product. In this situation, we collect data from all field sites. For data that requires manual collection, take a small random sample of field sites.

We collect data on failure intensity and on customer satisfaction with the major quality characteristics and use this information in setting the failure intensity objective for the next release. We also measure operational profiles in the field and use this information to correct the operational profiles we estimated. Finally, we collect information that will let us refine the process of choosing reliability strategies in future projects.

10.12. Conclusion

If you apply SRE in all the software-based products you develop, you will be controlling the process rather than it controlling you. You will find that you can be confident of the reliability and availability of the products. At the same time, you will deliver them in minimum time and cost for those levels of reliability and availability. You will have maximized your efficiency in satisfying your customers' needs. This is a vital skill to possess if you are to be competitive in today's marketplace.

11. Software Testing

11.1. Introduction

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we cannot completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time and quality.

Software Testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Software is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails. Most physical systems fail in a fixed (and reasonably small) set of ways. By contrast, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible. Unlike most physical systems, most of the defects in software are design errors, not manufacturing defects. Software does not suffer from corrosion, wear-and-tear -- generally it will not change until upgrades, or until obsolescence. So once the software is shipped, the design defects -- or bugs -- will be buried in and remain latent until activation.

Software bugs will almost always exist in any software module with moderate size: not because programmers are careless or irresponsible, but because the complexity of software is generally intractable -- and humans have only limited ability to manage complexity. It is also true that for any complex systems, design defects can never be completely ruled out. Discovering the design defects in software, is equally difficult, for the same reason of complexity. Because software and any digital systems

are not continuous, testing boundary values are not sufficient to guarantee correctness. All the possible values need to be tested and verified, but complete testing is infeasible. Exhaustively testing a simple program to add only two integer inputs of 32-bits (yielding 2^64 distinct test cases) would take hundreds of years, even if tests were performed at a rate of thousands per second. Obviously, for a realistic software module, the complexity can be far beyond the example mentioned here. If inputs from the real world are involved, the problem will get worse, because timing and unpredictable environmental effects and human interactions are all possible input parameters under consideration.

A further complication has to do with the dynamic nature of programs. If a failure occurs during preliminary testing and the code is changed, the software may now work for a test case that it didn't work for previously. But its behavior on pre-error test cases that it passed before can no longer be guaranteed. To account for this possibility, testing should be restarted. The expense of doing this is often prohibitive.

Regardless of the limitations, testing is an integral part in software development. It is broadly deployed in every phase in the software development cycle. Typically, more than 50% percent of the development time is spent in testing. Testing is usually performed for the following purposes:

- To improve quality

As computers and software are used in critical applications, the outcome of a bug can be severe. Bugs can cause huge losses. Bugs in critical systems have caused airplane crashes, allowed space shuttle missions to go awry, halted trading on the stock market, and worse. Bugs can kill. Bugs can cause disasters. The so-called year 2000 (Y2K) bug has given birth to a cottage industry of consultants and programming tools dedicated to making sure the modern world doesn't come to a screeching halt on the first day of the next century. In a computerized embedded world, the quality and reliability of software is a matter of life and death.

Quality means the conformance to the specified design requirement. Being correct, the minimum requirement of quality, means performing as required under specified circumstances. Debugging, a narrow view of software testing, is performed heavily to find out design defects by the programmer. The imperfection of human

nature makes it almost impossible to make a moderately complex program correct the first time. Finding the problems and get them fixed, is the purpose of debugging in programming phase.

- For Verification & Validation (V&V)

Just as topic indicated above, another important purpose of testing is verification and validation (V&V). Testing can serve as metrics. It is heavily used as a tool in the V&V process. Testers can make claims based on interpretations of the testing results, which either the product works under certain situations, or it does not work. We can also compare the quality among different products under the same specification, based on results from the same test.

We cannot test quality directly, but we can test related factors to make quality visible. Quality has three sets of factors -- functionality, engineering, and adaptability. These three sets of factors can be thought of as dimensions in the software quality space. Each dimension may be broken down into its component factors and considerations at successively lower levels of detail. Table 11.1 illustrates some of the most frequently cited quality considerations.

Table 11.1: Typical Software Quality Factors

Functionality (exterior quality)	**Engineering (interior quality)**	**Adaptability (future quality)**
Correctness	Efficiency	Flexibility
Reliability	Testability	Reusability
Usability	Documentation	Maintainability
Integrity	Structure	

Good testing provides measures for all relevant factors. The importance of any particular factor varies from application to application. Any system where human lives are at stake must place extreme emphasis on reliability and integrity. In the typical business system usability and maintainability are the key factors, while for a one-time scientific program neither may be significant. Our testing, to be fully effective, must be geared to measuring each relevant factor and thus forcing quality to become tangible and visible. Tests with the purpose of validating the product works are named clean tests, or positive tests. The drawbacks are that it can only validate that the software works for the specified test cases. A finite number of tests cannot validate that the software works for all situations. On the contrary, only one

failed test is sufficient enough to show that the software does not work. Dirty tests, or negative tests, refer to the tests aiming at breaking the software, or showing that it does not work. A piece of software must have sufficient exception handling capabilities to survive a significant level of dirty tests. A testable design is a design that can be easily validated, falsified and maintained. Because testing is a rigorous effort and requires significant time and cost, design for testability is also an important design rule for software development.

- For reliability estimation

Software reliability has important relations with many aspects of software, including the structure, and the amount of testing it has been subjected to. Based on an operational profile (an estimate of the relative frequency of use of various inputs to the program), testing can serve as a statistical sampling method to gain failure data for reliability estimation.

Software testing is not mature. It still remains an art, because we still cannot make it a science. We are still using the same testing techniques invented 20-30 years ago, some of which are crafted methods or heuristics rather than good engineering methods. Software testing can be costly, but not testing software is even more expensive, especially in places that human lives are at stake. Solving the software-testing problem is no easier than solving the Turing halting problem. We can never be sure that a piece of software is correct. We can never be sure that the specifications are correct. No verification system can verify every correct program. We can never be certain that a verification system is correct either.

11.2. Key Concepts

11.2.1. Correctness Testing

- Black-box testing

The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It is also termed data-driven, input/output driven, or requirements-based testing. Because only the functionality of the software module is of concern, black-box testing also mainly refers to functional testing -- a testing method emphasized on executing the

functions and examination of their input and output data. The tester treats the software under test as a black box -- only the inputs, outputs and specification are visible, and the functionality is determined by observing the outputs to corresponding inputs. In testing, various inputs are exercised and the outputs are compared against specification to validate the correctness. All test cases are derived from the specification. No implementation details of the code are considered.

It is obvious that the more we have covered in the input space, the more problems we will find and therefore we will be more confident about the quality of the software. Ideally we would be tempted to exhaustively test the input space. But as stated above, exhaustively testing the combinations of valid inputs will be impossible for most of the programs, let alone considering invalid inputs, timing, sequence, and resource variables. Combinatorial explosion is the major roadblock in functional testing. To make things worse, we can never be sure whether the specification is either correct or complete. Due to limitations of the language used in the specifications (usually natural language), ambiguity is often inevitable. Even if we use some type of formal or restricted language, we may still fail to write down all the possible cases in the specification. Sometimes, the specification itself becomes an intractable problem: it is not possible to specify precisely every situation that can be encountered using limited words. And people can seldom specify clearly what they want -- they usually can tell whether a prototype is, or is not, what they want after they have been finished. Specification problems contribute approximately 30 percent of all bugs in software.

The research in black-box testing mainly focuses on how to maximize the effectiveness of testing with minimum cost, usually the number of test cases. It is not possible to exhaust the input space, but it is possible to exhaustively test a subset of the input space. Partitioning is one of the common techniques. If we have partitioned the input space and assume all the input values in a partition is equivalent, then we only need to test one representative value in each partition to sufficiently cover the whole input space. Domain testing partitions the input domain into regions, and consider the input values in each domain an equivalent class. Domains can be exhaustively tested and covered by selecting a representative value(s) in each domain. Boundary values are of special interest. Experience shows that test cases that explore boundary conditions have a higher payoff than test cases that do not. Boundary value

analysis requires one or more boundary values selected as representative test cases. The difficulties with domain testing are that incorrect domain definitions in the specification cannot be efficiently discovered.

Good partitioning requires knowledge of the software structure. A good testing plan will not only contain black-box testing, but also white-box approaches, and combinations of the two.

- White-box testing

Contrary to black box testing, software is viewed as a white-box, or glass-box in white-box testing, as the structure and flow of the software under test are visible to the tester. Testing plans are made according to the details of the software implementation, such as programming language, logic, and styles. Test cases are derived from the program structure. White-box testing is also called glass-box testing, logic-driven testing or design-based testing.

There are many techniques available in white-box testing, because the problem of intractability is eased by specific knowledge and attention on the structure of the software under test. The intention of exhausting some aspect of the software is still strong in white-box testing, and some degree of exhaustion can be achieved, such as executing each line of code at least once (statement coverage), traverse every branch statements (branch coverage), or cover all the possible combinations of true and false condition predicates (Multiple condition coverage).

Control-flow testing, loop testing, and data-flow testing, all maps the corresponding flow structure of the software into a directed graph. Test cases are carefully selected based on the criterion that all the nodes or paths are covered or traversed at least once. By doing so we may discover unnecessary "dead" code -- code that is of no use, or never get executed at all, which cannot be discovered by functional testing.

In mutation testing, the original program code is perturbed and many mutated programs are created, each contains one fault. Each faulty version of the program is called a mutant. Test data are selected based on the effectiveness of failing the mutants. The more mutants a test case can kill, the better the test case is considered. The problem with mutation testing is that it is too computationally expensive to use.

The boundary between black-box approach and white-box approach is not clear-cut. Many testing strategies mentioned above, may not be safely classified into black-box testing or white-box testing. It is also true for transaction-flow testing, syntax testing, finite-state testing, and many other testing strategies not discussed in this text. One reason is that all the above techniques will need some knowledge of the specification of the software under test. Another reason is that the idea of specification itself is broad -- it may contain any requirement including the structure, programming language, and programming style as part of the specification content.

We may be reluctant to consider random testing as a testing technique. The test case selection is simple and straightforward: they are randomly chosen. Study in indicates that random testing is more cost effective for many programs. Some very subtle errors can be discovered with low cost. And it is also not inferior in coverage than other carefully designed testing techniques. One can also obtain reliability estimate using random testing results based on operational profiles. Effectively combining random testing with other testing techniques may yield more powerful and cost-effective testing strategies.

11.2.2. Performance testing

Not all software systems have specifications on performance explicitly. But every system will have implicit performance requirements. The software should not take infinite time or infinite resource to execute. "Performance bugs" sometimes are used to refer to those design problems in software that cause the system performance to degrade.

Performance has always been a great concern and a driving force of computer evolution. Performance evaluation of a software system usually includes: resource usage, throughput, stimulus-response time and queue lengths detailing the average or maximum number of tasks waiting to be serviced by selected resources. Typical resources that need to be considered include network bandwidth requirements, CPU cycles, disk space, disk access operations, and memory usage. The goal of performance testing can be performance bottleneck identification, performance comparison and evaluation, etc. The typical method of doing performance testing is using a benchmark -- a program, workload or trace designed to be representative of the typical system usage.

11.2.3. Reliability testing

Software reliability refers to the probability of failure-free operation of a system. It is related to many aspects of software, including the testing process. Directly estimating software reliability by quantifying its related factors can be difficult. Testing is an effective sampling method to measure software reliability. Guided by the operational profile, software testing (usually black-box testing) can be used to obtain failure data, and an estimation model can be further used to analyze the data to estimate the present reliability and predict future reliability. Therefore, based on the estimation, the developers can decide whether to release the software, and the users can decide whether to adopt and use the software. Risk of using software can also be assessed based on reliability information advocates that the primary goal of testing should be to measure the dependability of tested software.

There is agreement on the intuitive meaning of dependable software: it does not fail in unexpected or catastrophic ways. Robustness testing and stress testing are variances of reliability testing based on this simple criterion.

The robustness of a software component is the degree to which it can function correctly in the presence of exceptional inputs or stressful environmental conditions. Robustness testing differs with correctness testing in the sense that the functional correctness of the software is not of concern. It only watches for robustness problems such as machine crashes, process hangs or abnormal termination. The oracle is relatively simple therefore robustness testing can be made more portable and scalable than correctness testing. This research has drawn more and more interests recently, most of which uses commercial operating systems as their target, such as the work in Stress testing, or load testing, is often used to test the whole system rather than the software alone. In such tests the software or system are exercised with or beyond the specified limits. Typical stress includes resource exhaustion, bursts of activities, and sustained high loads.

11.2.4. Security testing

Software quality, reliability and security are tightly coupled. Flaws in software can be exploited by intruders to open security holes. With the development of the Internet, software security problems are becoming even more severe. Many critical software applications and services have integrated security measures against malicious attacks.

The purpose of security testing of these systems include identifying and removing software flaws that may potentially lead to security violations, and validating the effectiveness of security measures. Simulated security attacks can be performed to find vulnerabilities.

11.3. Testing Automation

Software testing can be very costly. Automation is a good way to cut down time and cost. Software testing tools and techniques usually suffer from a lack of generic applicability and scalability. The reason is straight-forward. In order to automate the process, we have to have some ways to generate oracles from the specification, and generate test cases to test the target software against the oracles to decide their correctness. Today we still don't have a full-scale system that has achieved this goal. In general, significant amount of human intervention is still needed in testing. The degree of automation remains at the automated test script level.

The problem is lessened in reliability testing and performance testing. In robustness testing, the simple specification and oracle: doesn't crash, doesn't hang suffices. Similar simple metrics can also be used in stress testing.

11.4. When to Stop Testing?

Testing is potentially endless. We cannot test till all the defects are unearthed and removed -- it is simply impossible. At some point, we have to stop testing and ship the software. The question is when. Realistically, testing is a trade-off between budget, time and quality. It is driven by profit models. The pessimistic, and unfortunately most often used approach is to stop testing whenever some, or any of the allocated resources -- time, budget, or test cases -- are exhausted. The optimistic stopping rule is to stop testing when either reliability meets the requirement, or the benefit from continuing testing cannot justify the testing cost. This will usually require the use of reliability models to evaluate and predict reliability of the software under test. Each evaluation requires repeated running of the following cycle: failure data gathering -- modeling -- prediction. This method does not fit well for ultra-dependable systems, however, because the real field failure data will take too long to accumulate.

11.5. Alternatives to Testing

Software testing is more and more considered a problematic method toward better quality. Using testing to locate and correct software defects can be an endless process. Bugs cannot be completely ruled out. Just as the complexity barrier indicates: chances are testing and fixing problems may not necessarily improve the quality and reliability of the software. Sometimes fixing a problem may introduce much more severe problems into the system, happened after bug fixes, such as the telephone outage in California and eastern seaboard in 1991. The disaster happened after changing 3 lines of code in the signaling system.

11.6. Verification/Validation/Certification

In the development of an embedded system, it is important to be able to determine if the system meets specifications and if its outputs are correct. This is the process of verification and validation (V & V) and its planning must start early in the development life cycle. Both aspects are necessary as a system meeting its specifications does not necessary mean it is technically correct and vice versa. There are many different V & V techniques, which are applicable at different stages of the development life cycle. The results of V & V forms an important component in the safety case, which is a document used to support certification. Certification is usually pursued due to either legal reasons or economic advantages. The certification process also starts from the beginning of the life cycle and requires cooperation between the developer and regulatory agency from the very start. Thorough V & V does not prove that the system is safe or dependable, and there is always a limit to how much testing is enough testing. In addition, certification does not prove that a system is correct, so it does not eliminate the developer's legal and moral obligations. Therefore, extreme care should be taken in the development of embedded systems to make sure that the right amount of time is spent on V & V, and also that certification not be used to prove that a system is correct.

Verification, validation, and certification are essential in the life cycle of any safety critical embedded system. The development of any system is not complete without rigorous testing and verification that the implementation is consistent with the specifications. Verification and validation (V & V) have become important, especially in software, as the complexity of software in systems has increased, and

planning for V & V is necessary from the beginning of the development life cycle. Over the past 20 to 30 years, software development has evolved from small tasks involving a few people to enormously large tasks involving a many people. Because of this change, verification and validation has similarly also undergone a change. Previously, verification and validation was an informal process performed by the software engineer himself. However, as the complexity of systems increased, it became obvious that continuing this type of testing would result in unreliable products. It became necessary to look at V & V as a separate activity in the overall software development life cycle. The V & V of today is significantly different from the past as it is practiced over the entire software life cycle. It is also highly formalized and sometimes activities are performed by organizations independent of the software developer. [Andriole86] In addition, V & V is very closely linked with certification because it is a major component in support of certification.

11.6.1. Verification Techniques

There are many different verification techniques but they all basically fall into 2 major categories - dynamic testing and static testing.

- **Dynamic testing** - Testing that involves the execution of a system or component. Basically, a number of test cases are chosen, where each test case consists of test data. These input test cases are used to determine output test results. Dynamic testing can be further divided into three categories - functional testing, structural testing, and random testing.
- **Functional testing** - Testing that involves identifying and testing all the functions of the system as defined within the requirements. This form of testing is an example of black-box testing since it involves no knowledge of the implementation of the system.
- **Structural testing** - Testing that has full knowledge of the implementation of the system and is an example of white-box testing. It uses the information from the internal structure of a system to devise tests to check the operation of individual components. Functional and structural testing both chooses test cases that investigate a particular characteristic of the system.
- **Random testing** - Testing that freely chooses test cases among the set of all possible test cases. The use of randomly determined inputs can detect faults

that go undetected by other systematic testing techniques. Exhaustive testing, where the input test cases consists of every possible set of input values, is a form of random testing. Although exhaustive testing performed at every stage in the life cycle results in a complete verification of the system, it is realistically impossible to accomplish. [Andriole86]

- **Static testing** - Testing that does not involve the operation of the system or component. Some of these techniques are performed manually while others are automated. Static testing can be further divided into 2 categories - techniques that analyze consistency and techniques that measure some program property.
- **Consistency techniques** - Techniques that are used to insure program properties such as correct syntax, correct parameter matching between procedures, correct typing, and correct requirements and specifications translation.

11.6.2. Validation Techniques

There are also numerous validation techniques, including formal methods, fault injection, and dependability analysis. Validation usually takes place at the end of the development cycle, and looks at the complete system as opposed to verification, which focuses on smaller sub-systems.

- **Formal methods** - Formal methods is not only a verification technique but also a validation technique. Formal methods means the use of mathematical and logical techniques to express, investigate, and analyze the specification, design, documentation, and behavior of both hardware and software.
- **Fault injection** - Fault injection is the intentional activation of faults by either hardware or software means to observe the system operation under fault conditions.
- **Hardware fault injection** - Can also be called physical fault injection because we are actually injecting faults into the physical hardware.
- **Software fault injection** - Errors are injected into the memory of the computer by software techniques. Software fault injection is basically a simulation of hardware fault injection.

- **Dependability analysis** - Dependability analysis involves identifying hazards and then proposing methods that reduces the risk of the hazard occurring.
- **Hazard analysis** - Involves using guidelines to identify hazards, their root causes, and possible countermeasures.
- **Risk analysis** - Takes hazard analysis further by identifying the possible consequences of each hazard and their probability of occurring.

Verification and validation is a very time consuming process as it consists of planning from the start, the development of test cases, the actual testing, and the analysis of the testing results. It is important that there are people specifically in charge of V & V that can work with the designers. Since exhaustive testing is not feasible for any complex system, an issue that occurs is how much testing is enough testing. Sure, the more testing the better but when do the cost and time of testing outweigh the advantages gained from testing. The amount of time and money spent on V & V will certainly vary from project to project. In many organizations, testing is done until either or both time and money runs out. Whether this method is effective or not, it is a technique used by many companies.

11.7. Certification Process

Verification and validation are part of the long certification process for any embedded system. There are different reasons why a product needs certification. Sometimes certification is required for legal reasons. For example, before an aircraft is allowed to fly, it must obtain a license. Being certified would also be important for commercial reasons like having a sales advantage. One of the main reasons for certification is to show competence in specific areas. Certification is usually carried out by government agencies or other organizations with a national standing.

Certification can be applied to organizations or individuals, tools or methods, or systems or products. Certification of organizations aims at assuring that the organization achieves a certain level or proficiency and those they agree to certain standards or criteria's. This however, is not applicable to all areas because while it is easy to measure the procedures of a company, it is much harder to measure the competence with which they are performed. So certification is usually applied to areas, such as quality assurance and testing, as opposed to design. Certification may

also apply to individuals where workers must be certified in order to be a certain profession. This usually applies to workers such as doctors, lawyers, accountants, and civil engineers. Tools or methods may also be certified. In certification, there is always the issue of whether artifacts or methodology be certified. This becomes an issue in the certification of products containing software. Because software testing is so difficult, certification must be based on the process of development and on the demonstrated performance. This is a case where the methodology (development process) is certified instead of the artifact (software).

Even though certification does not occur until the end of a system development cycle, the planning starts from the very beginning. Because certification is a complicated process between the developer and the regulatory agency, the certification liaison between the parties must be established early on in the process. Next, the developer should submit a verification plan for approval by the regulatory agency. After the submission, discussion takes place between the developer and regulatory agency to resolve areas of misunderstanding and disagreement. Changes to the methods used have to be approved by the regulatory body to insure that certification will not be affected. Throughout the entire development life cycle of the product, documentation must be continually submitted to show that the certification plan is satisfied. The regulating authority will also hold a series of reviews to discuss the submitted material. At the end, if the terms of the certification plan have been satisfied, then a certificate or license is issued.

The safety case is an important document used to support certification. It contains a set of arguments supported by analytical and experimental evidence concerning the safety of a design. It is created early in the development cycle and is then expanded as issues important to safety come up. In the safety case, the regulatory authority will look to see that all potential hazards have been identified, and that appropriate steps have been taken to deal with them. In addition, the safety case must also demonstrate that appropriate development methods have been adopted and that they have been performed correctly. Items that should be included in the safety case includes, but are not limited to the following: specification of safety requirements, results of hazard and risk analysis, verification and validation strategy, and results of all verification and validation activities.

A potential problem with certification is that manufacturers use it to avoid its legal and moral obligations. An important aspect of certification is that it does not prove that the system is correct. Certification only proves that a system has met certain standards set by the certifying agency. The standards show that a product has met certain guidelines, but it does not mean that the system is correct. Any problem with the system is ultimately the responsibility of the designer and manufacturer, not the certification agency.

11.8. Test Planning

System testing is expensive. Careful planning is needed to get the most out of testing and to control testing costs. Test planning is concerned with setting out standards for the testing process rather than describing product tests. They allow technical staff to get an overall picture of the system tests and to place their own work in this context.

Unit testing and module testing may be the responsibility of the programmers developing the component. Programmers make up their own test data and incrementally test the code as it is developed. This is an economically sensible approach as the programmer knows the component best and is most able to generate test data but it is a natural human trait for individuals to feel an affinity with objects they have constructed and programmers may feel that testing threatens their creations. If unit testing is left to the component developer, it should be re-tested by independent tester or should be subjected to some other monitoring procedure.

Later stages of testing involve integration work from a number of programmers and must be planned in advance. An independent team of testers should undertake them. Module and sub-system should be planned as the design of the sub-system is formulated. Integration tests should be developed in conjunction with the system design whereas Acceptance tests should be designed with the program specifications.

11.9. Statistical Testing

In order to test program's performance and reliability, tests are designed to reflect the frequency of actual user inputs. After running the tests, an estimate of the operational reliability of the system can be made. Program performance may be judged by measuring the execution of the statistical tests.

11.10. Defect Testing

When defects have been found in a program, these must be discovered and removed. This is called debugging. Testing establishes the existence of defects. Debugging is concerned with locating and correcting these defects. Defect Testing is intended to find areas where the program does not conform to its specification.

11.11. Stages in Testing Process

Execution based software testing, especially for large systems are usually carried out at different levels. In most cases there will be 3-4 levels, or major phases of testing; unit test, integration test, system test and some type of acceptance test. At each level there are specific testing goals. For example, at unit test a single component is tested. A principal goal is to detect functional and structural defects in the unit. At the integration level several components are tested as a group and the tester investigates component interactions. At the system level the system as a whole is tested and a principle goal is to evaluate attributes such as reliability, usability and performance. Software developed for the mass market (i.e. shrink-wrapped software) often goes through a series of tests called alpha and beta tests. Alpha tests bring potential users to the developer's site to use the software. Developers note any problems. Beta tests send the software out to potential users who use it under real-world conditions and report defects to the developing organization.

11.11.1. Unit Testing

Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

11.11.2. Module Testing

A module is a collection of dependant components such as an object class, an abstract data type or some looser collection of procedures and functions. A module encapsulates related components so can be tested without other system modules.

11.11.3. Sub-System Testing

This phase involves testing collection of modules, which have been integrated into sub-systems. Sub-systems may be independently designed and implemented. The most common problems, which arise in large software systems, are sub-system

interface mismatches. The sub-system test process therefore should concentrate on detection of interface errors by rigorously exercising these interfaces.

11.11.4. System Testing

The subsystems are integrated to make up the entire system. The testing process is concerned with finding errors, which result from unanticipated interactions between subsystems and system components. It is also concerned with validating that the system meets its functional and non-functional requirements.

11.11.5. Acceptance Testing

This is the final stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the system procurer rather than simulated test data. Acceptance testing may reveal errors and omissions in the system requirements definition because the real data exercises the system in different ways from the test data. Acceptance testing may also reveal requirement problems where the system's facilities do not really meet the user's needs or the system's performance is unacceptable. Acceptance testing is sometimes called ALPHA Testing. Alpha testing process continues until the system developer and the client agree that the delivered system is an acceptable implementation of system's requirement.

11.11.6. Beta Testing

When a system is to be marketed as a software product the testing process used is called BETA testing. Beta testing involves delivering the system to number of potential customers who agree to use that system. They report problems to the system developers. This exposes the product to real use and detects errors, which may not have been anticipated by system builders. After the feedback the system is modified and either released for further Beta testing or for general sale.

11.12. Comparative Review of Testing Strategies

A testing strategy is a general approach to the testing process rather than a method of devising particular system or component tests. Different testing strategies may be adopted depending on the type of system to be tested and the development process used.

11.12.1. Top Down Testing

Testing starts with the most abstract component and works downwards. Top Down Testing tests the high levels of a system before testing its detailed components. The program is represented as a single abstract component with sub-components represented by stubs. Stubs have the same interface as the component but a very limited functionality. Top Down Testing should be used with Top down program development so that a system component is tested as soon as it is coded. Coding and testing are a single activity with no separate component or module-testing phase

Advantages: If top down testing is used, unnoticed design errors may be detected at an early stage in the testing process. As these errors are usually structural errors, early detection means that they can be corrected without undue costs. Early error detection means that extensive re-design and re-implementation may be avoided. Working system is available at an early stage in the development.

Disadvantages: Strict top-down testing is difficult to implement because of the requirement that program stubs, simulating lower levels of the system, must be produced. These program stubs may either be implemented as a simplified version of the component required which returns some random value of the correct type or by manual simulation. If the component is a complex one, it may be impractical to produce a program stub which simulates it correctly. Test output may be difficult to observe. In many systems, the higher levels of that system do not generate output but, to test these levels, they must be forced to do so. The tester may create an artificial environment to generate the test results. Collection of objects is not usually integrated in a strictly hierarchical way so a strict top-down testing strategy is not appropriate for object-oriented systems.

11.12.2. Bottom Up Testing

In this approach testing starts with the fundamental components and works upwards. It involves testing the modules at the lower levels in the hierarchy, and then working up the hierarchy of modules until the final module is tested. When using bottom-up testing, test drivers must be written to exercise the lower level components. These test drivers simulate the components' environment and are valuable components in their own right.

Advantages: Disadvantages of top-down testing are the advantages of bottom-up testing. Bottom-up testing is appropriate for object-oriented systems in that individual objects may be tested using their own test drivers. Bottom-up testing of critical, low-level system components is almost always necessary.

Disadvantages: If top-down development is combined with bottom-up testing, all parts of the system must be implemented before testing can begin. Architectural faults are unlikely to be discovered until much of the system has been tested. Correction of these faults might involve the rewriting and consequent re-testing of lower level modules in the system.

11.12.3. Thread Testing

Thread Testing is used for systems' with multiple processes where the processing of a transaction threads its way through these processes. Testing strategy devised for testing real time systems. It is an event- based approach where tests are based on events which trigger system actions.

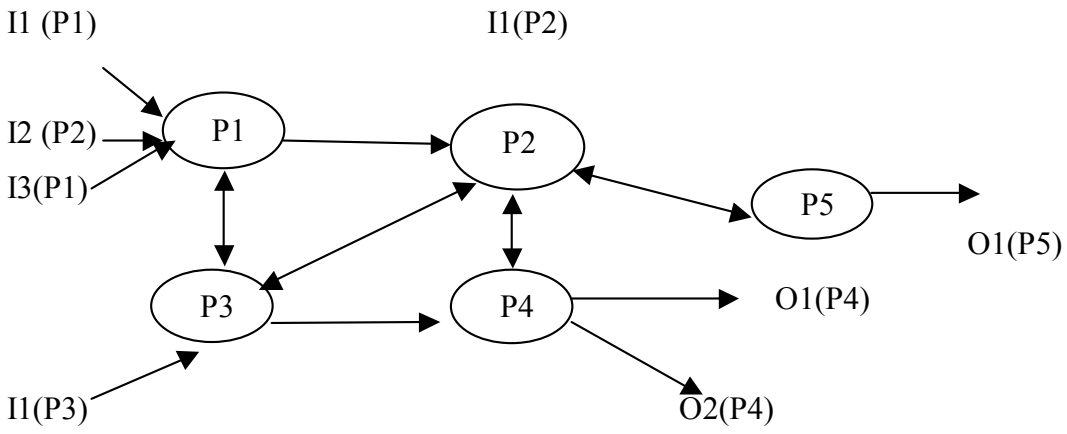


Figure 11.1: Thread Testing

Thread testing is testing strategy, which may be used after processes, or objects have been individually tested and integrated into sub-systems. Thread testing involves identifying and executing each possible processing thread.

Because of practical difficulty in complete thread testing, only the most commonly exercised threads can be identified and selected for testing. After each thread has been tested with a single event, the processing of multiple events of the same type should be tested without events of any other type. After the system's

reaction to each class of event has been tested, it can then be tested for its reactions to more than one class of simultaneous event.

Advantages: Thread testing can be most beneficially used after processes or objects have been individually tested and integrated into sub-systems.

Disadvantages: In real-rime systems it may be very difficult to identify all the threads. Further, it may be impossible to execute each possible thread.

11.12.4. Stress Testing

It involves planning a series of tests where the load is steadily increased. Stress testing continues these tests beyond the maximum design load of the system until the system fails. Stress testing is particularly relevant to distributed systems based on a network of processors because these systems often exhibit severe degradation when they are heavily loaded as the network becomes swamped with data which the different processes must exchange.

1. It tests the failure behavior of the system. Stress testing checks that overloading the system causes it to fail-soft rather than collapse under its load.
2. It stresses the system and may cause defects to come to light which would not normally manifest themselves.

Advantages: Stress testing checks that overloading the system causes it to fail-soft rather than collapse under its load. It is important to see that system failure does not cause data corruption or unexpected loss of user services. Stress testing simulates unusual combinations of normal circumstances which lead the system to failure.

***Disadvantages*:** It is generally felt that the defects simulated by stress test are unlikely to cause system failures in normal usage because such unusual combinations resulting in high stress may never occur in reality.

11.12.5. Back-to-Back Testing

Back to back testing is only possible in the following situations:-

3. When a system prototype is available.

4. When reliable systems are developed using N-version programming.
5. When different versions of a system have been developed for different type of computers.

Steps in back-to-back testing

6. Prepare a general-purpose set of test cases.
7. Run one version of the program with these test cases and save the results in one or more files.
8. Run another version of the program with the same test cases, saving the results to a different file.
9. Automatically compare the files produced by the modified and unmodified program versions.

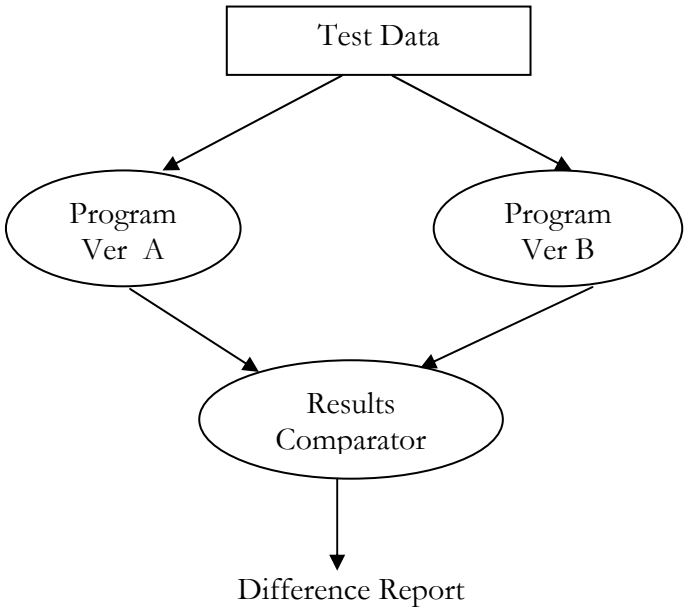


Figure 11.2: Back to back testing

Differences between the outputs suggest problems which should be investigated in more details.

***Advantages*:** It is a very easy method of comparing the results when two versions of a system or system prototype is available.

Disadvantages: Back-to-back testing may not be always possible since it is not usually realistic to generate a completely new system only for testing. Further, if the

file comparison shows the output files to be identical, it does not guarantee that they are valid since the implementers of both versions may have made the same mistake.

11.13. Comparative Review of Defect Testing Approaches

Defect testing demonstrates the presence not the absence of program faults. This contrasts with Validation testing which is intended to demonstrate that a system meets its specifications.

It is practically impossible for defect testing to be exhaustive. Test cases in this

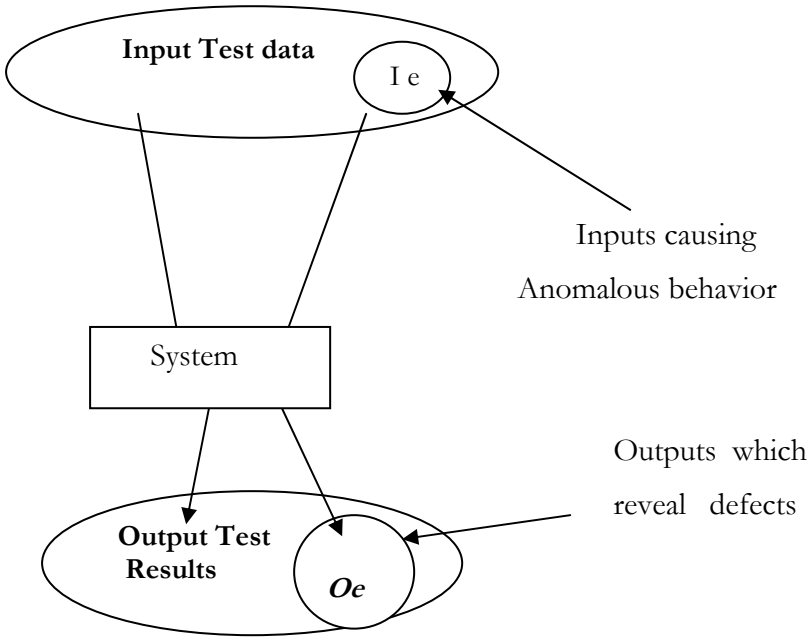


Figure 11.3: Black box testing

case need to be chosen such that faults which disrupt the operation are detected before the defects which just have nuisance value. If a program is a revision of an existing system, it can be expected that existing features must be working well so new features can be tested first. Typical situation testing can be done prior to boundary value cases.

11.13.1. Functional or Black-box testing

In this approach the tests are derived from the program specification. The system is a black box whose behavior can only be determined by studying its inputs and the related outputs.

Black box testing relies on the specification of the system or component which is being tested to derive test cases. The key problem for the defect tester is to select inputs that have a high probability of being members of the set Ie. And in many cases the selection of these test cases is based on the previous experience of test engineers.

***Advantages*:** As per the experiment conducted by Basili and Selby in 1987, it was demonstrated that Black box testing was more effective in discovering faults than structural testing. Further, it is easy to conduct Black box testing since the knowledge of program's structure and implementation is not essential.

***Disadvantages*:** It may not be always reasonable to surmise that if the test fails to detect defects when one member of a class is processed, no other members of the class would identify defects. Further, some equivalence partitions may not be identified or errors may be made in equivalence partition identification or the test data may be incorrectly prepared. These tests do not check for unexpected corruption of data outside the component.

11.13.2. Structural or White-box testing

In white box testing, tests are derived from the knowledge of program's structure and implementation. It is also called Glass box testing. Analysis of code can be used to find how many test cases are needed to guarantee a given level of test coverage. Knowledge of algorithm used to implement some function can be used to identify further equivalence partitions. One method of white box testing is path testing.

Path Testing is white-box testing strategy where objective is to exercise every independent execution path through the component. Starting point for path testing is a program flow graph.

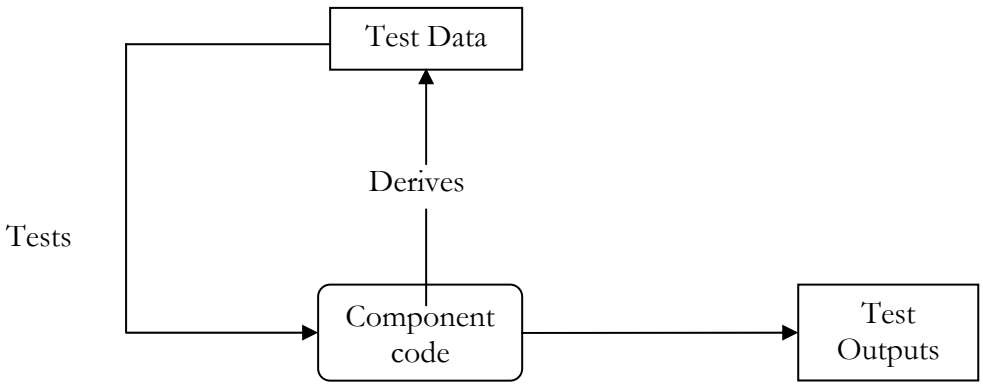


Figure 11.4: White-box Testing

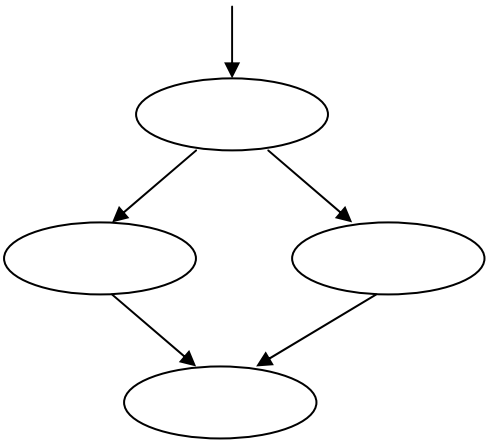


Figure 11.5: Flow Graph: If-then-else

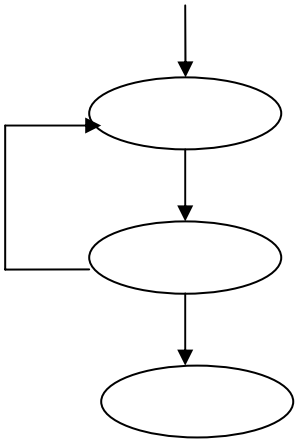


Figure 11.6: Flow Graph: While loop

***Advantages*:** Analysis of the code can be used to find how many test cases are needed to guarantee a given level of test coverage. Knowledge of the algorithm used to implement some function can be used to identify further equivalence partitions.

Disadvantages: Path testing used in this technique may not test all possible combinations of all paths through the program. For any components apart from very trivial ones without loops, this is an impossible objective. Further, the number of paths through a program is usually proportional to its size. As modules are integrated into systems, it becomes unfeasible to use structural testing techniques. Path testing techniques are therefore only really usable at the unit testing and module testing stages of the testing processes.

11.13.3. Interface Testing

Tests are derived from the program specification plus knowledge of its internal interfaces. This type of testing is particularly important for Object-oriented systems. The objective is to detect faults which may have been introduced into the system because of interface errors or invalid assumptions about interfaces.

Different types of interfaces are:

Parameter interfaces: Where data or sometimes function references are passed from one component to another.

Shared memory interfaces: Interfaces where a block of memory is shared between sub-systems. Data is placed in the memory by one subsystem and retrieved from there by other sub-systems.

Procedural interfaces: Interfaces where one sub-system encapsulates a set of procedures which can be called by other sub-systems. Objects and abstract data types have this form of interface.

Message passing interfaces: Interface where one sub-system requests a service from another sub-system by passing a message to it.

***Interface Errors*:** Interface errors are one of the most common forms of error in complex systems. These errors fall into three classes:

Interface misuse: A calling component calls some other component and makes an error in the use of its interface.

Interface misunderstanding: A calling component misunderstands the specification of the interface of the called components and embeds assumptions about the behavior of the called component. The called component does not behave as expected and this causes unexpected behavior in the calling component.

Timing errors: These occur in real-time systems which use a shared memory or a message passing interface. The producer of data and consumer of data may operate at different speeds. Unless particular care is taken in the interface design, the consumer can access out-of-date information because the producer of the information has not updated the shared interface information.

Guidelines for Interface testing

- Examine the code and design a set of tests (extreme ends of their ranges).
- Where pointers are passed across an interface, always test the interface with null pointer parameters.
- Design tests which should cause the component to fail.
- Use a stress testing strategy in message passing systems to reveal timing problems.

Design tests that vary the order in which several components are activated to reveal implicit assumptions made by the programmer about the order in which the shared data is produced and consumed.

***Advantages*:** Interface testing is very useful in finding faults in large and complex systems since Interface errors are one of the most common forms of error in complex systems.

***Disadvantages*:** Testing for interface defects is particularly difficult because interface faults may only manifest themselves under unusual conditions. Because of interactions between faults in different modules or objects, faults in one object may only be detected when some other object behaves in an unexpected way. Many interface errors may be detected by compiler of strongly typed language leaving very little need to do interface testing.

11.14. Conclusions

- Software testing is an art. Most of the testing methods and practices are not very different from 20 years ago. It is nowhere near maturity, although there are many tools and techniques available to use. Good testing also requires a tester's creativity, experience and intuition, together with proper techniques.
- Testing is more than just debugging. Testing is not only used to locate defects and correct them. It is also used in validation, verification process, and reliability measurement.
- Testing is expensive. Automation is a good way to cut down cost and time. Testing efficiency and effectiveness is the criteria for coverage-based testing techniques.
- Complete testing is infeasible. Complexity is the root of the problem. At some point, software testing has to be stopped and product has to be shipped. The stopping time can be decided by the trade-off of time and budget. Or if the reliability estimate of the software product meets requirement.

12. Field Data Analysis

12.1. Introduction

The role and functionality of software in modern computer-based systems is growing at a tremendous rate. At the same time, pressures are mounting on software developers to deliver software of better quality, and to actively monitor the field performance of their software. Current experience indicates that software failures are increasing in proportion to system failures as organizations create more complex systems, while the information about these failures is frequently less than complete, uniform or precise. For example, field data on large telephone switching systems indicate that software is responsible for 20% to 50% of complete system outages. Figure 12.1 illustrates the percentage of reported causes of total system outages (due to hardware, software, and other causes) for two large telecommunications systems. The values indicated are averaged over several releases. Although both systems have similar overall functionality, there are some remarkable differences that underlie important, and often observed, property of software field reliability data — their variability.

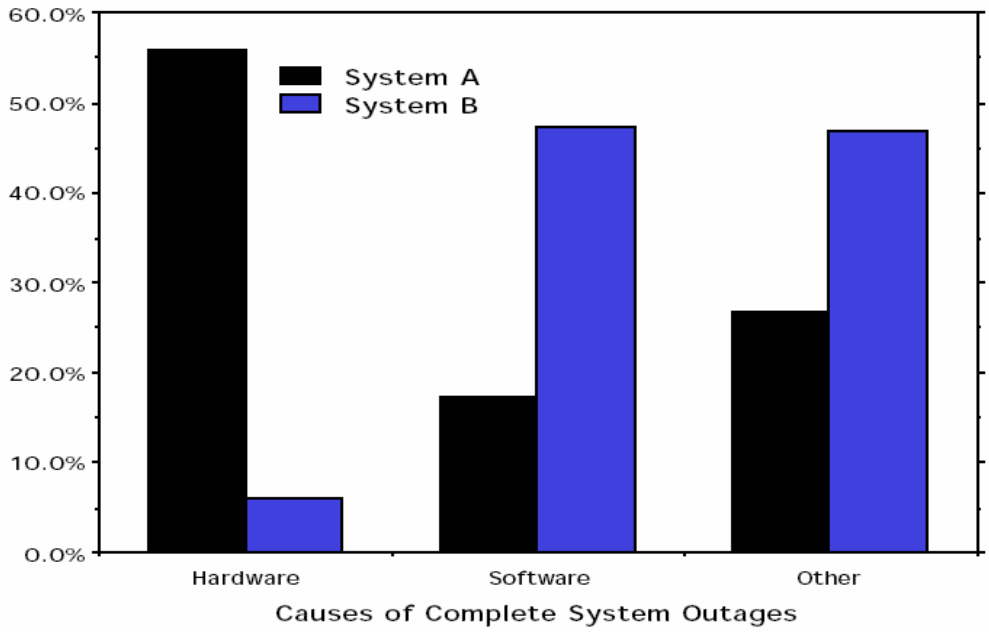


Figure 12.1: Causes of complete system outages averaged over several releases for two large telecommunication systems: System A and System B.

When examining individual releases for System A, about 30%-60% of outages were attributed to hardware (some of which may have involved a combination of hardware and software problems), about 20%-25% were attributed to software, and procedural and other errors accounted for the remainder of the outages. In the case of System B 3%-7% of outages were attributed to hardware, and between 15% and 60% (depending on the maturity of the release) could be attributed to software. The figures reported for System A are closer to the distributions reported for operating systems. The variance between System A and B is due to, at least in part, the lack of a precise definition for software outage categories. It may also differ due to the functional implementation strategy of the two systems (for example, System A may implement more functionality in hardware). Whatever the reasons, it is not easy to compare the two systems and draw objective conclusions.

Examples like the one above can be found in all application areas. Therefore, it is not surprising that there are industrial, national and international efforts to standardize software reliability data collection and analysis processes. For example, in the U.S., Bellcore is an organization that acts as a software quality "watchdog" from within the telecommunications community. Bellcore requires collection of outage data for all network switching elements, analysis of the data, and classification of the data by cause of failure.

A proper collection and analysis of *software* failure data lies at the heart of a practical evaluation of the quality of software-based systems. Software-based systems differ from pure hardware systems in many ways. Software failures are not driven by the physical wear-out seen in hardware, and software repair processes and procedures are different than those for hardware. Furthermore, in practice, software is more change-prone than hardware. As a result, studies of software data have their own unique data collection and analysis requirements apart from hardware.

These requirements become even more important when we consider analysis of software field data as opposed to test data. There is usually much less control over what is actually collected in the field, often analyses are based on the available historical data, and operational usage of the system usually cannot be stopped to await the analysis of the data. In addition, organizations are much more sensitive to disclosure of field data due to competitive pressures.

It is possible for an organization to spend considerable resources on collecting software field failure data with minimal returns as there may be no clear understanding as to why particular data are collected or how the data are to be analyzed. This is one of the reasons why studies in software reliability must have clearly defined objectives, goals, and analysis methods so that efficient use may be made of the existing data, and that the cost of collecting required supplemental data is minimized.

The goal of this chapter is to provide insight into the process of collection and analysis of software reliability field data through a discussion of the underlying principles and case study illustrations. A distinction is made between the data collected from the actual operational sites where software is used by its intended users during field tests or in day-to-day production, and the data collected during *controlled* system tests and experiments with operational software. The latter categories were discussed in the earlier chapters, and therefore are not considered in this chapter.

12.2. Data Collection Principles

Software reliability is often expressed in terms of probability of failure in a given time, or in terms of the failure intensity, which is the number of failures per unit time. Minimum data requirements for calculating one expression may be slightly different than the other. Furthermore, precision in the data collection mechanism may affect the variance in reliability parameter estimates or field predictions. The basic information required to perform reliability analyses includes the amount of time a software system is in operation and the exact times that failures occur. A less precise, but usable, alternative would be condensed data that only reports the total number of failures observed over a period of time. Also, additional data may be required if we wish to do more than analyze the reliability of the product. For example, if we desire to determine the availability of the product, we need both failure repair and failure severity information.

12.2.1. Study Plans, Goals and Input Variables

The data needed for collection and its subsequent analysis (yielding information) should be related to the **goals** of the study. In reliability field data analysis, some important goals are:

1. To assess the actual quality and reliability of a software product in its operational environment (which in turn assists in determining compliance with requirements or regulations and with the planning of maintenance resources),
2. To relate field failure behavior of software to its usage in the field, and to its development and maintenance processes, through models,
3. To predict software behavior in the field, and control its field quality by controlling its development, testing and maintenance processes and methods.

Currently, in industry the first goal has preeminence and is the logical first step when conducting field analysis. It also illustrates how.

Although various organizations may have different goals, exact and detailed goals are needed to properly carry out a study, or any other software related task. The first step is to develop a **study plan**. The plan details the goals, the deliverable, the methods, the processes, the schedules, the available resources, etc. For example, we may wish to determine whether availability of a given software release improves over time. To answer this question, we need to collect, and later select, failure and repair data for all sites that run this particular software version over the time period of interest. This subset of data, called the **study population**, is defined by the study goals, methods and deliverables. Unless the scope of study goals is defined well in advance, so that the study input data **variables** are within that scope, the desired analyses may not be possible and incomplete or wrong information may be collected instead. The study plan is a living document that the study should follow. The plan should be regularly updated to account for any changes and for the feedback from the process itself and from the study.

12.2.2. Failures, Faults, and Related Data

Accurate field collection of this information and related data is essential to any serious software reliability engineering effort. In addition to the recording of the failures and the times of the corrective actions, there is other information that is helpful for a full. Table 12.1 provides an example of the data that can help a designer take corrective action, and also allow an analyst to properly segment and prepare data for system-level software reliability analysis. In the table, we distinguish between

general classifiers, such as date and time of failure, and software-specific classifiers, such as software version information and causal analysis information.

Table 12.1: Examples of fields required for reliability analysis

Note	General Classifiers	Example
Required	Date failure occurred	921214
	Time failure occurred	045600
	Date failure was reported	921214
	Tracking number or identifier (It often helps to make these identifiers as informative as possible)	ATCH-E2-1-00076
	Customer name or code	American Technology
Recommended	Site code, or a comparable entity	CHCGILAA34F
	Customer severity of failure (for example, Critical-High-Medium-Low)	High
	Degradation	
	level of degradation to system (in percent)	5
	duration of degradation	23
	Apparent cause - top level classification (determined at time of screening, e.g., hardware, software,	ISDN Call Processing
	Root cause (to be determined later by vendor)	Table Control and Config
	System or subsystem level of failure	ISDN
	Status in Investigation (Not under inv., Under inv., Closed, Resolved)	Resolved
	Problem resolution process	
	Problem owner	J. Doe
	Status of problem (Open-Fixed-Rejected-Resolved)	Resolved
	Is the fix available? (Y or N)	Y
	Is this problem a duplicate of previous one? (Y or N) or the ID# of the duplicate	N
	Resolution date	921220
Software-Specific		
Required	Software Version	8.1
Recommended	Version of Software in underlying operating system	4.0
	Problem at install ? (Y or N)	N
	Failure type (Executable or Data)	Executable
	High Level Cause design or correction fault if Executable design or procedural fault if Data	Design-Logic
	Text describing the failure or the input state that reproduces the failure	ISDN PRI Trunks do not come up on warm restart when ...
	Patch created (Y or N)	Y
	Patch process	
	Patch Identifier	P-0192-064
	Status of patch (D-documented, C-coded, T-tested, A-available, GA-generally available)	GA
	Date patch is created	921221
	Version patch is written for	8.0+

The information in Table 12.1 would be drawn from a variety of sources: customers, field support personnel, problem screeners, designers, system engineers, and maintenance including patch applicators. However, it would be very difficult for a reliability analyst to gather this information individually for all failures. Instead, what is needed is a toolset that allows integration of information (whether preexisting or current) from many sources and a variety of forms (e.g., reports, files, or databases) so that an analyst can create a table such as Table 12.1.

The decisions on what data to collect, how to collect the data (for example, automated vs. manual) and how to **verify correctness** of the collected information, are some of the most crucial decisions an organization makes in its software reliability engineering program. Therefore they should be given appropriate attention and visibility.

Partnering with customers is essential. Without the customer's assistance it is very difficult to collect adequate field data for system analysis. The customers should know why the data is needed, how the data will be used, and how they will benefit from the analysis. Providing feedback to the customer regarding the information that is gleaned from customer field data is of great importance. It will enhance the quality of the data collected and provide customer focus that leads to quality improvement.

12.2.3. Time

In general, the more often that a product is used, the more likely that a failure will be experienced. A full implementation of software reliability engineering requires consideration of software usage through determination of *operational profile*(s), and analysis of observed problems in that context. For example, if a software subsystem (or module) is found to exhibit an excessive number of field problems, it should be established whether this is due to very frequent usage of a component that has an average residual fault density (perhaps expressed as number of faults per line of code), or due to an excessive residual fault density in a component that is being used at the rate typical for most other product components. Re-engineering of both subsystems may be required. However, the evaluation of the process that created each subsystem would be very different. For example, the first subsystem may have an issue in understanding the demanding requirements that are associated with highly

utilized components and not necessarily the implementation quality. The first subsystem may also need more extensive verification than the second.

"Time" is the execution exposure that the software receives through usage. As experience indicates that the best measure of time is the actual central processing unit (CPU) execution-time ([Musa87]). However, CPU time may not be available, and it is often possible to reformulate the measurements, and *reliability models*, in terms of other exposure metrics: calendar-time, clock time, in-service time (usually a sum of clock times due to many software applications running simultaneously on various single or multiple CPU systems), logical time (such as number of executed test cases, or fraction of planned test cases executed), or structural coverage (such as branch achieved statement or branch coverage). In service time usually implies that each system is treated as one unit whether it has one or several CPUs. Also, 100 months of in-service time may be associated with 50 months (clock time) of two systems or 1 month (clock time) of 100 systems. In many cases, in service time like clock time will be proportional to system execution (CPU) time. For this chapter, the term "usage time" will refer to any of CPU, execution, clock, or in service time.

12.2.4. Usage

Ideally, one should have a record of everywhere the system is used, and some information on how it is used. This type of information allows calculation of metrics such as the total number of systems in operation on a given date and total operation time accumulated over all licensed systems running a particular version of the software.

Some operating systems support collection of usage data better than others. For example, processes can be created in UNIX that allow tracking of when the software is accessed, who accesses it, how frequently it is accessed, and how long the user accesses it. This allows collection of usage data at the CPU level. However, to do this in a thorough manner, exact knowledge of the users (through licenses and other means) is often necessary, as is access to the user's system.

12.2.5. Data Granularity

In collecting usage and other data it should be remembered that the useful precision of the estimate/prediction of reliability is always less than the precision of the data.

For example, predictions for how many failures will occur during a particular week will be of little use if the data is only collected monthly. Therefore, choosing the right granularity is very important. For example, time interval for data sampling or aggregation may be one second, one hour, one day, one week, one month, ten test cases, one structural branch, or some other value. The time granularity of the raw data determines the lower limits of meaningful micro modeling and analyses that can be performed.

For a different illustration, consider prediction of the time-to-next failure, a standard metric in reliability analysis. With field data, predicting the time-to-next failure or even the next five failures is usually impractical. In many cases when field data is assimilated for analysis, groups of failures (say 5-10 in size) are commonly associated with the same time frame (say one calendar week). Predicting that the next failure will occur within the next ten usage weeks with a probability of 0.95 will not help the customer since ten usage weeks may correspond to three calendar days. Thus, by the time all the data has been collected and analyzed, the next failure has *already* occurred. Field usage is very different from the laboratory test environment where one can interrupt the testing and assess the reliability of the system before continuing with another round of tests. Field usage is continuous; therefore, analysis should be commensurate with practical data collection delays and should focus on longer-range forecasting and estimation since this can be adequately done even when the failure and/or usage data is "lumped" together.

12.2.6. Data Maintenance and Validation

In practice, a large amount of failure data may be entered manually by field support personnel from customer reports or interviews. Some software systems have internal or independent mechanisms that detect failures of various types, and record that data automatically for later retrieval and processing. Even if such an automated system is in place, some data may still need to be entered manually simply because the data entry program either cannot function during a failure state or cannot recognize all failure states. Furthermore, some automated systems often cannot distinguish between hardware and software failures, and thus manual identification is required. Nevertheless, for any system, information surrounding a failure needs to be recorded as accurately as possible and data entry and database systems should be designed in such a way that all of the pertinent information is available to a reliability analyst.

Automation of date and time entries, implementation of intra- and inter-data record error and consistency checking, and standardization of entries will ensure that the analyst will have the best data possible from which to draw information. The database that holds the field data must be updated and crosschecked as new data becomes available or as existing data is found to be inaccurate. The importance of consistency checking cannot be overstressed. Unfortunately, it is an area that most data collection systems overlook. The effects of data discrepancies can be very pronounced, especially in the early deployment life when the usage data is sparse. For example, even a relatively small mistake in accounting for the sites involved, or in associating failures with the appropriate software releases, can have considerable impact on the failure count and the computation of the failure intensity.

12.2.7. Analysis Environments

For the proper analysis, many pieces are required that must work well together. First, there must be processes and tools in place to collect the raw data. There must also be an appropriate storage mechanism for the data, which is usually a database. If the database does not allow for easy data scanning, manipulation, and processing, then some other system should be in place to allow cursory examination and filtering of inappropriate or corrupt data. Of course, corrupt data should be corrected if possible, or at least marked as such. After filtering, an environment for merging data from different sources should be in place, since the data needed for failure analysis often reside in different systems. Also, some data may need to be transformed. Finally, for modeling and estimation, an environment that supports statistical methods should be available as well as a good data graphing tool. Depending on how the data will be used in a given environment, various information feedback mechanisms may be needed for different job roles that utilize that information.

Reliability analysts typically require access to thousands of records simultaneously, and they have interests in not only aggregating the data, but in examining what historical data is available. For example, an analyst may wish to know whether or not a database can produce an historical image of itself at some given previous point in time. This determines whether or not an organization could simulate historical events and data available at these events and thus enable a "prediction" from strictly old data.

12.3. Data Analysis Principles

In statistics, analysis of data is usually considered exploratory or confirmatory. **Exploratory** analysis includes techniques in which one is only beginning to conjecture associations and the objective is simply to explore the potential nature of the data. **Confirmatory** techniques are typically used after some body of evidence has emerged to confirm or challenge the prevailing wisdom or current thought. The **hypothesis test** is a tool very frequently used in confirmatory analysis. Although hypothesis test results are published in exploratory analysis studies, the nature of the study usually violates assumptions that artificially inflate the statistical significance of the study, or the hypothesis is often a straw man that is easily "crumpled" by the weight of the data.

There are several exploratory data analysis techniques that are particularly relevant in the analysis of software failure and fault data. They are *plots and graphs*, *data modeling* and associated diagnostics, *data transformation*, and *data resistance*. Each technique has its own special utility but can often be used in combination with each other.

It is often assumed that the field software exhibits reliability growth. However, this assumption needs to be validated in each study. There are two primary reasons for the assumption of reliability growth. First, most software systems can be patched relatively easily. In fact, patching is one of the great advantages of software over hardware. Faults are corrected while the system is in operation, and the system subsequently experiences reliability growth. Second, users of the system may become "familiar" with the imminent-failure states through first-hand experience, or information from other users, or the vendor. This information tends to allow the user to avoid failure modes until a correction occurs.

In the following sub-sections we will examine various elementary data analysis principles. We present the ideas using field data from a real large release of software from a major digital telecommunications company. The data set, called Dataset 1 and is given in the Appendix A. In most cases, we will be concerned with reliability growth models, although most of the techniques we discuss will apply to a variety of other models and analyses.

12.3.1. Plots and Graphs

Plots and graphs are very powerful tools in exploratory analysis, particularly when coupled with color graphics. It is often the case that an analyst can determine very quickly the initial relationships and associations in a data set using scatter plots, line plots, stem-and-leaf plots, and schematic or box plots. In software reliability, one often sees plots of the main variables of interest. For example, for Dataset 1 the line plot in Figure 12.2 illustrates the relationship between the total number of sites using the software release related to Dataset 1 (version N) and calendar time. We see that the number of offices is initially low, but quickly "ramps" up to the point of saturation. After the next release becomes available, the number of offices having version N steadily declines as customers migrate to the new release N+1. This graph illustrates that the usage of version N is far from constant in calendar time, an important factor to consider when examining the reliability of this software since usage often will not be proportional to calendar time.

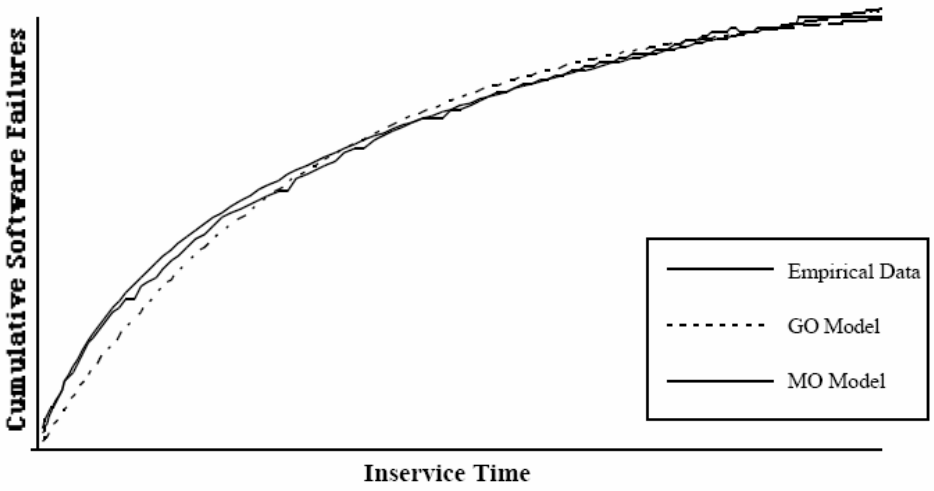


Figure 12.2: Two potential functional relationships: MO Model and GO Model

Another frequently used graph in software reliability illustrates the relationship between cumulative software failures and usage time. For example, Bellcore as a mandatory graph stipulates this graph that U.S. telecommunications suppliers must provide in their Reliability and Quality Measurement System reports. Figure 12.3 is an example of this graph for system Dataset 1. Note that the data has been normalized to protect proprietary information. The main effect of normalization on the analysis is one of scaling. Therefore, the analysis of the non-normalized data would be essentially the same.

Based on Figure 12.3, we may conjecture that some simple functional relationship may exist between cumulative failures and time. In fact, two potential functional relationships are shown in Figure 12.3 using models. If both models appear to "fit" or "describe" the data equally well then you have encountered the unfortunate limitations of perception with curved graphs. It is very difficult to distinguish one type of curve from another. The "fitted" curves are actually very different functions; one is a logarithmic function and the other is an exponential function. Therefore, the moral is *do not use* cumulative failure plots to determine functional relationships, or compare different functional relationships. Although either model may be useful for interpolation, it is extrapolation (or predictions) of behavior that is of primary interest to a reliability engineer. These two functions have vastly different extrapolations. Thus, graphs of the cumulative failures should in practice be limited to depicting the failures for a given release against a predicted curve, or in simultaneously comparing several releases in an overlay plot.

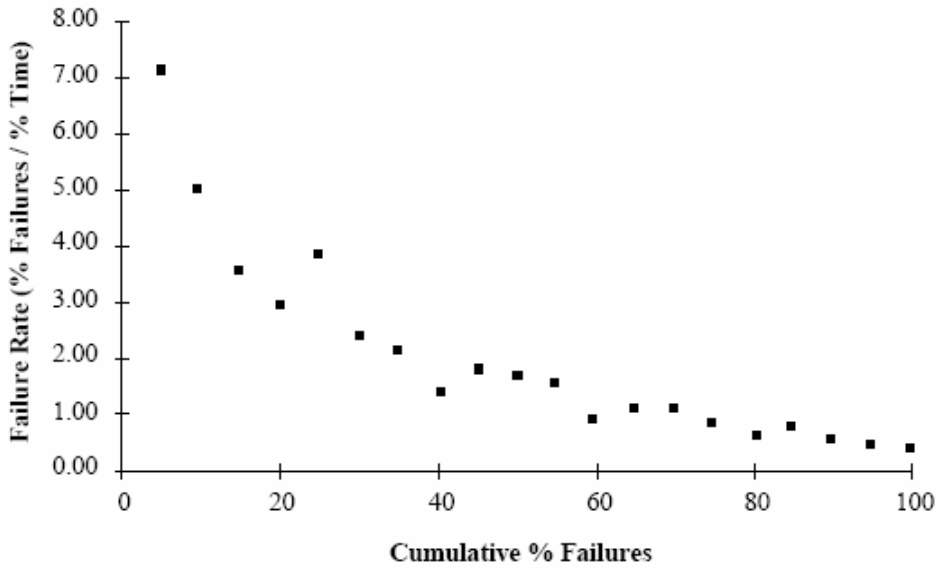


Figure 12.3: Scatter plot of the failure intensity of Data-Set 1. Time is measured in in-service units.

Failure intensity is the rate of change in the expected cumulative failures. The number of failures per unit time can quantify it. Since the failure intensity changes over time, we are interested in the instantaneous failure intensity and how it changes with respect to time, or how it changes with accumulation of failures. Figure 12.4 is a scatter plot of the failure intensity of release Data Set 1 with a group size of 5

(percent) 1 against the cumulative failure count (in this case normalized to the total number of recorded failures).

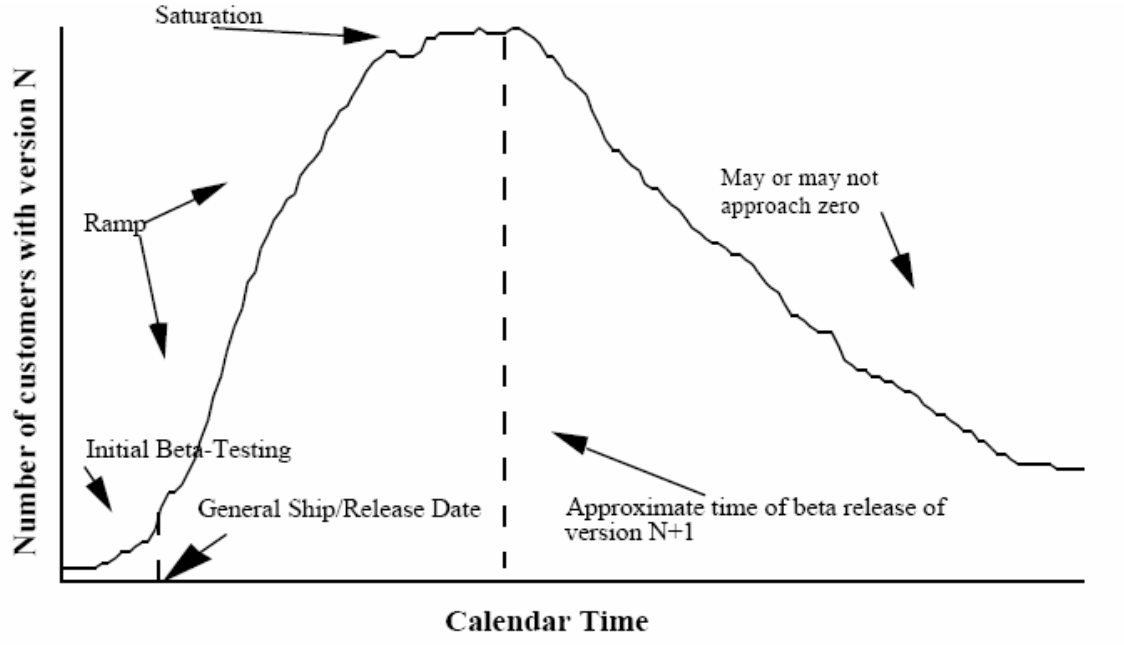


Figure 12.4: The "loading" or installation "ramping" effect (Dataset 1).

Failure intensity *should* play an important role in any reliability analysis. Many of the graphs illustrated in this text (including Figure 12.4), and many graphical diagnostics, require calculation of the approximate failure intensity from empirical data. This calculation has many benefits: the empirical failure intensity can be measured and quantified, graphs of the failure intensity may indicate appropriate functions, and parameters for certain models may be successfully estimated from the empirical failure intensity using ordinary least squares (in addition to more complex estimation methods such as maximum likelihood). Inspection of Figure 12.4 reveals that the failure intensity appears to decrease (indicating reliability growth) in a non-linear fashion, and that the variance in the failure intensity becomes smaller as it approaches zero. The obvious and uniform decreasing trend exhibited in Figure 12.4 may not be as obvious in other situations. For example, immediately after initial deployment of a release (during the so called "transient" region where the usage load is low and small errors in the data can drastically affect all metrics, including failure intensity), or where the data has large variance, we would like to confirm that reliability growth actually occurs before we commit to a particular (global) reliability growth model.

12.3.2. Data Modeling and Diagnostics

Models are very important to engineers. Most useful models are predictive, and some models may be used to direct development and maintenance process management. We will assume, for convenience, that software failures occur in accordance within the general framework of the non-homogeneous Poisson process (NHPP). In principle, this assumption which underlies many of the models, should be confirmed before we attempt to fit these models to data. The test requires information on true inter-failure times, something that may be difficult to obtain for the field data.

Table 12.2: Review of some well-known software reliability growth models in $\mu(\tau)$ and $\lambda(\tau)$

Model	Mean-Value Function	Failure Intensity Function in Time	Initial Intensity
Goel-Okumoto (GO) Basic Exponential	$\mu(\tau) = \beta(1 - \exp(-\alpha\tau))$	$\lambda(\tau) = \alpha\beta\exp(-\alpha\tau)$	$\alpha\beta$
Musa-Okumoto (MO) Logarithmic Poisson	$\mu(\tau) = \frac{1}{\theta}\ln(\lambda_0\theta\tau + 1)$	$\lambda(\tau) = \frac{\lambda_0}{\lambda_0\theta\tau + 1}$	λ_0
Littlewood-Verrall	$\mu(\tau) = \frac{1}{\beta_1}\left[-\beta_0 + \sqrt{\beta_0^2 + 2\beta_1\tau}\right]$	$\lambda(\tau) = \frac{1}{\sqrt{\beta_0^2 + 2\beta_1\tau}}$	$\frac{1}{\beta_0}$
Duane (Crow)	$\mu(\tau) = \beta\tau^{\alpha}, \alpha < 1$	$\lambda(\tau) = \alpha\beta\tau^{\alpha-1}$	undef.

The NHPP framework is very flexible and is not limited by specific assumptions that were common with initial models (for example, the assumption of instantaneous perfect repair of faults or the total number of failures is constant but unknown). It also allows for the use of covariates in the mean-value function that may or may not be directly tied with usage time. Table 12.2 provides a review of some well-known Software Reliability Growth Models in mean value function, $\mu(\tau)$, and failure intensity, $\lambda(\tau)$, that will be important to our discussion of diagnostics.

12.4. Important Topics in Analysis of Field Data

In the case of a multi-release system, at different calendar times different software releases are installed at a different number of sites. This means that the usage intensity of a particular software load varies over calendar time and accumulates usage according to the amount of time the sites using the release have been in

service. Therefore, from both hardware and software viewpoints, in-service time is a more representative and relevant measure of usage than calendar time. However, in many cases, calendar time is a measure that better reflects the perception of users (such as the telecommunications companies) since calendar time availability and degradation of services are very important from the customers' point of view. Simply stated, the context may dictate whether one or both viewpoints (calendar time and in-service time) are appropriate for analysis.

When discussing the quality of software in operational use, it is instructive to employ a classification based on the usage characteristics of the product and the nature and availability of the field data. As seen in previous chapters, it is well known that the software usage profile is a dominant factor that influences the reliability, and that the software execution time is a better time domain than calendar time since it automatically incorporates the workload to which the software is subjected. The influence of measuring usage on reliability modeling as an alternative to calendar time is demonstrated by the example in this section. However, in practice, we may have to make a statement about the quality of the software without having direct information about its usage, and without having available a large number of failure events. Therefore, in the following sections, we will discuss three classes of field reliability data analysis: calendar-time, usage-time and special event analysis. In the last section, we will discuss the related concept of availability.

12.4.1. Calendar Time

Calendar-time analysis arises in situations where failures are reported only in the calendar-time domain and precise information about the usage of the software may not be available. We see this type of constraint in wide-distribution software — software that is developed for the purpose of selling on the open market to many customers, or for non-profit distribution to anyone who wishes to install it. Its usage often builds to thousands, or even hundreds of thousands, of independent systems. However, direct monitoring of the usage rate of such software is not always feasible, or is not practiced. This is especially true of commercial wide-distribution software, shrink-wrapped or off-the-shelf software, and freeware. Examples of wide-distribution commercial software are Microsoft Windows, WordPerfect, DEC's Ultrix, commercial PC and workstation compilers, and freeware such as the GNU family of software products

12.4.2. Usage Time

It should not be surprising that the majority of the organizations that are prominent in the practice of software reliability engineering (SRE) deal with telecommunications and safety-critical applications. Other application areas include reservation systems, banking transaction systems, database engines, operating systems, medical instruments, etc. For many of these systems, reliability is one of the most important, if not the most important, attribute of the system. This implies the need for accurate and detailed information about the system usage.

Usage-time analysis can be performed when more precise information about software usage is available. This is often true for software that is developed for the purpose of selling to a specialized market such as the examples given above. Its usage may build up to hundreds or thousands of independent systems, yet the users of the software are known and well documented, and direct monitoring of the usage rate of the software is feasible and is practiced.

12.4.3. An Example

The following example helps underscore the issues driving the above classification. Figure 12.5 shows the actual field data for a large-scale limited-distribution telecommunications product. We plot the concurrent changes in the number of installed systems of a particular software release (Dataset 1 given in Appendix A) over calendar time, the corresponding failure rate in terms of calendar time (failures per week), and failure rate per system in-service week. Note the dramatic difference between the failures per calendar week and the failures per system in-service week.

From Figure 12.5, we see that the calendar-time failure rate is initially low (indicating apparent high reliability), then begins to climb (apparent reliability degradation), and finally reaches a peak just before the deployment reaches its peak. A naive analyst might mistakenly conclude that a disaster is in the making. In fact, the system is behaving as it should — the problem is an inherent deficiency in the failure rate metric. As the rate of deployment peaks, the reliability appears to improve dramatically, and the failure rate drops steadily thereafter. However, we see a different picture when we examine the failure rate in terms of failures per system in-service week (that is, normalized with respect to the deployment function) or per usage load on the system. The normalized failure rate is initially high, but then

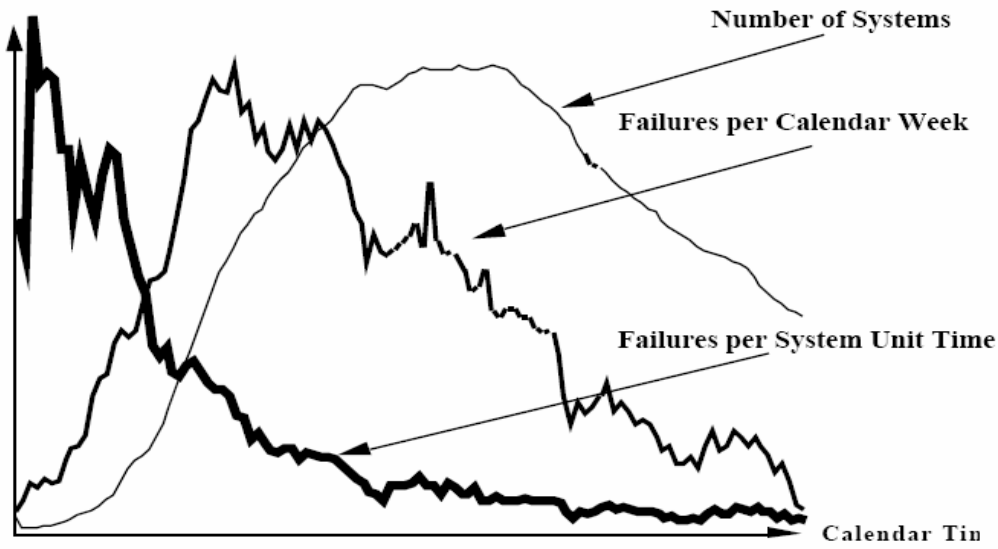


Figure 12.5: Influence of usage on failure

decreases dramatically in the first few weeks after the system has been deployed. As the deployment curve peaks, the reliability growth may slow but reliability continues to improve. Obviously, the model that describes the failure behavior of this system will depend very strongly on whether we have the actual system usage information or not. The number of failures per calendar week is a direct function of the true reliability of the system, *and* the deployment function of the system. Reliability growth may be difficult, if not impossible, to discern from the calendar based view. While failures per calendar week may be a natural, and important, metric to a customer service organization, it is usually far from suitable for making inferences about the reliability of the system [Musa87].

In other cases, the number of failures (in this case, failures which cause outages) may track the number of deployed systems closely indicating no real reliability growth even though the number of failures diminishes with calendar time. This is illustrated in Figure 12.6 using real data for a release of a large telecommunications system. A similar relationship was observed for many other versions of the same product. This suggests that in the situations when usage information may be inaccurate or unavailable, failure counts over a lengthier period of time may offer a valid and useful measure of software quality, especially if usage remains relatively unchanged from release to release.

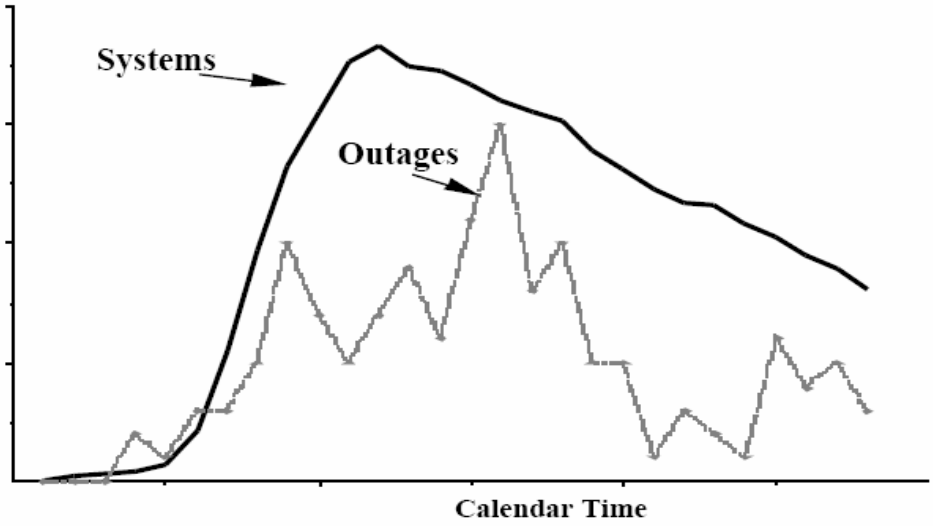


Figure 12.6: Calendar time-dependence of observed outages and of the number of installed systems for a large release of telecommunications software.

12.5. Calendar-Time Reliability Analysis

The principal characteristics of wide-distribution software are that it is used by many users at many customer sites. This software lives in the world of multiple releases, and for this type of software, by definition, we often do not know who the users are or how they use it. Although we may know how many licenses there are, we may not know how much each copy is used. Software for single-user systems is purchased and installed, but sometimes rarely, if ever, used. Software for multi-user systems may have up to thousands of users. In fact, with site licensing, we often do not know how many copies of the software are being used. When dealing with wide-distribution commercial software we often have a large user base with each user experiencing his/her own level of reliability. Some users may run for months without a disruption, while others may only run for a few hours before running into a problem. It all depends on the user's software usage profile. Yet, despite the fact that reliability of a software system is important to customers of commercial software products, they generally do not keep good records on execution time and they seldom report on their reliability experience. What they do report to software development organizations is the occurrence of specific failures, with the expectation of getting the underlying defect(s) fixed so that the failures do not reoccur. This is possibly why many commercial software development organizations focus on the number of remaining defects rather than reliability, or mean time to failure, as the measure of software quality. Musa et al. discuss the advantages and disadvantages of the

calendar-based analysis in great detail, and show that although a general Weibull-type failure intensity model can describe calendar time system behavior, the fit is often inferior to the one obtained using execution-time based intensity [Musa87]. But, they also point out that in practice managers and users may be more attached to the calendar-time domain since it is closer to the world in which they make decisions.

12.6. Usage-Based Reliability Analysis

As vendors and customers form closer alliances due to stringent reliability expectations, usage data will be more accessible. Currently, Bellcore requires that telecommunication vendors in the United States systematically record software usage and a variety of metrics that quantify the quality of software releases. In fact, it is likely that in the near future, industries such as medical software will be subject to similar requirements from some outside agency. In this section, we show how software reliability analysis can be conducted when sufficient usage information is available. We use recent examples from government and industry. We will see that although one model may suit a particular system very well (even over several releases), there is no model which is optimal across all systems. Different models are used with different systems due to the nature of the system, changes in the failure process, and the needs of the study of the system. This finding amplifies the need to conduct, for the system of interest, analysis and model selection and validation in a scientific and systematic way so that an adequate model and method is found. Model re-evaluation and validation should occur in intervals that will depend on the stability of software usage profiles and failure processes.

12.7. Special Events

Some classes of failures may be of special interest, and may be considered more important than others. Usually these are failures that are categorized as having life-threatening, or extremely damaging consequences. The need to recognize early the potential of a software-based system for special event problems is obvious. How to achieve this is less clear. In general, it is necessary to link the symptoms observed during, say, software testing phases with the effects observed in the operational phase. In that context, the key is identification of these failure modes, and of the events that lead to these failures. Failures modes that are absolutely unacceptable should **not** be analyzed using only probabilistic methods since these methods are

inherently incapable of assuring the level of reliability that is required for such systems. Some other techniques, such as formal methods, should be used to complement the analyses. Ultra-high reliability systems pose special problems and require dependability assessment techniques beyond the scope of this chapter. A good discussion of these issues can be found in and.

However, special event failures to which one is willing to attach a probability of occurrence (say, above 10-7) may be analyzable through the concept of risk. This concept forms a bridge between the probabilistic reliability aspects and the critical and economic considerations of any system. A risk model identifies a set of software reliability engineering indicators or symptoms, and relates them to the expected behavior of the software in the field.

An example of a special event that could be regarded through the probabilistic prism is an FCC-reportable failure. In part owing to a series of operational problems that have embarrassed switching industry in the past several years4, FCC has issued a notification to common carriers regarding service disruptions that exceed 30 minutes and affect more than 50,000 lines. Since March 1992 any outage of this type needs to be reported to FCC within 30 minutes of its occurrence. These FCC-reportable events are relatively rare, but such outages may have serious safety implications5. Since the information itself can command considerable public visibility and attention, such failures may have serious business implications as well.

12.7.1. Rare Event Models

The key issue is the probability of occurrence of rare events. Computation of the probability of rare software events is not a solved problem, and perhaps not even a fully solvable problem. However, whatever results are available must be presented not as a point estimate but as a range, or interval. For example [lower bound, upper bound]. Often 95% confidence interval is used. We present some very simple models which serve to highlight the issues involved, and indicate the difficulty of the problem.

12.7.1.1. Constant Failure-Rate Model

If some failure information is available, and it can be assumed that the failure rate, or failure intensity, is constant, then one deals with a Gamma (exponential) distribution

when the problem is such that the number of failures is fixed, but the total exposure time is a random variable, or the Poisson distribution when the number of failures is a random variable, but the total exposure time is fixed. In that case, standard statistical confidence bounds for these distributions can be used to evaluate the information. The simplest model is the one where we estimate the probability of the undesirable events based on the counts of these events:

$$P(S_f) \simeq \frac{n_f}{n}$$

Where, n_f is the number of failure events, and n is the usage exposure expressed as the number of intervals in which we wish to estimate.

12.7.1.2. Reliability Growth

If the usage rate of the product is growing, but its quality remains approximately the same, or grows at a lower rate than the product usage, then although per site failure rate may be roughly constant (or may even be improving), the overall number of reported problems will grow. In that case, it is necessary to model the per site failure rate. For example, let function S(t) describe the number of sites that use a particular release of a product at some calendar-time t (see Figure 12.4 and Figure 12.5). This shape can often be described using a Poisson [Leve90], or perhaps Weibull-type envelope, such as

$$s(t) = K \frac{\alpha}{\beta} t^{\alpha-1} e^{-(\frac{t}{\beta})^{\alpha}}$$

Combined with historical information about the "usual" position of the envelope mode, and other model parameters, and the marketing information about a release, e.g., the total number of sites expected to run this release, it may be possible to predict the site profiles relatively early in the life-cycle of a release. Summation of over all active releases can then yield the overall load on the software release. If this function is then combined with the one describing the quality of the release, it may be possible to make early and accurate predictions of the outage rates.

12.8. Availability

An important concept that is related to reliability is software availability. The importance stems from prevalent industry specifications related to reliability and availability. For example, one of Bellcore's primary requirements is for the availability of telecommunications a network element. Their target is about 3 minutes of downtime per year. Availability is simply the probability that the system will be available when demanded, and it depends on both the reliability and the reparability of the system.

12.8.1. Measuring Availability

12.8.1.1. Instantaneous Availability

Instantaneous availability is the probability that the system will be available at any random time t during its. We estimate "instantaneous" availability in a period i as follows:

$$\hat{A}(i) = \frac{\text{uptime in period i}}{\text{total inservice time for period i}}$$

Where, the in-service time is the total time in the period i during which all hosts of a particular type (e.g., DEC, SUN, processor A), at all sites, operated a particular software release (whether fully operational, partly degraded, or under repair), while uptime is the total time during period i at which the systems were not in the "100% down" state (or total system outage state). Correspondingly, the instantaneous unavailability estimate is (1 - ^A(i)). Associated with this measure are "instantaneous" system failure, $\lambda(i)$, and recovery rates, $\rho(i)$, which are estimated as follows:

$$\hat{\lambda}(i) = \frac{\text{number of outages in period i}}{\text{total uptime for period i}}$$

$$\frac{\text{number of outages in period i}}{\text{total downtime for period i}}$$

Where, "in-service time" for period i is the sum of the downtime and uptime in that period.

12.8.1.2. Average Availability

Since the raw data are often "noisy", the data are usually presented after some form of smoothing, or "data aging", has been applied. This gives rise to a family of "smoothed" availability metrics (there is, in fact, an analogous family of smoothed reliability metrics). Examples are one-sided moving average and symmetrical moving average, such as 11-point symmetrical moving average. An extreme form of smoothing is provided by the **average**, or **uptime**, availability. Uptime availability is the proportion of time in a specified interval [0,T] that the system is available for use

We estimate uptime availability up to and including period i as follows:

$$A_C(i) = \frac{\sum_{x=1}^{i} \text{uptime in period } x}{\sum_{x=1}^{i} \text{total inservice time for period } x}$$

Total uptime and total in service time are cumulative sums starting with the first observation related to a particular release. Uptime includes degraded service. Associated with uptime availability are average system failure, $\hat{\lambda}\, c(i)$, and recovery rates, $\hat{\rho}\, c(i)$, which are estimated as follows:

$$\hat{\lambda}\, c(i) = \frac{\sum_{x=1}^{i} \text{number of outages in period } x}{\sum_{x=1}^{i} \text{uptime in period } x}$$

$$\hat{\rho}\, c(i) = \frac{\sum_{x=1}^{i} \text{number of outages in period } x}{\sum_{x=1}^{i} \text{downtime in period } x}$$

12.8.2. Failure and Recovery Rates

Two measures which directly influence the availability of a system are its failure rate and its field repair rate (or software recovery rate). Figure 12.5 shows P2 failure and recovery rates for release R11. Apart from the censored13 "raw" data two other representations are shown. In one, the data are smoothed using an 11-point symmetrical moving average. In the other, we show cumulative average of the data.

In a system which improves with field usage we would expect a decreasing function for failure rate with in-service time (implying fault or problem reduction and reliability growth). Immediately after the product release date, there is considerable variation in the failure rate. Later the failure rate reduces and stabilizes.

Failure rate is connected to both the operational usage profile and the process of problem resolution and correction. Recovery rate depends on the operational usage profile, the type of problem encountered, and the field response to that problem (i.e., the duration of outages in this case). If the failures encountered during the operational phase of the release do not exhibit durations which would be preferentially longer or shorter at a point (or period) in the life-cycle, then we would expect the "instantaneous" recovery rate to be a level function with in-service time (with, perhaps, some oscillations in the early stages).

12.8.3. Models

The time varying nature of both the failure rate and, to a lesser extent, the repair rate indicates that a full availability model should be non-homogeneous. In addition, the distribution of outage causes, as well as the possibility of operation in degraded states, suggest that a detailed model should be a many-state model. Nonetheless, a very simple two-state model may provide a reasonable description of the system availability beyond the transient region.

It can be shown that system availability A(t) and unavailability– A(t) = 1 - A(t), given some simplifying assumptions, is:

$$A(t) = \frac{\rho}{\lambda+\rho} + \frac{\lambda}{\lambda+\rho}e^{-(\lambda+\rho)t}$$

It can also be shown that uptime availability can be formulated as,

$$A_c(T) = \frac{\rho}{\lambda+\rho} + \frac{\lambda}{(\lambda+\rho)^2 T} - \frac{\lambda}{(\lambda+\rho)^2 T}e^{-(\lambda+\rho)T}$$

The system becomes independent of its starting state after operating for enough time for the transient part of the above equations to decay away. This steady-state availability of the system is $A(\infty) = \text{limit}\{A(t = T \rightarrow \infty)\}$, i.e.,

$$A(\infty) = \frac{\rho}{\lambda+\rho}$$

The two-state model discussed above represents a system which can be either fully operational or completely off-line and under repair. However, not all realistic systems follow this simple model. In fact, systems like the ones discussed in our case studies not only have failure rates and repair rates which vary with time, and can have different down states, but they can also function in more than one up state (i.e., the system may remain operational but with less than 100% functionality for some failures). Thus, a many-state non-homogeneous Markov model may be more appropriate for describing the details of these systems.

12.8.4. Prediction

In practice, a model would be used to predict future unavailability of a system. Of course, only the data up to the point from which the prediction is being made would be available. We will refer to this point at which the prediction is made as the data "cut-off point". The prediction of future unavailability will differ from the true value depending on how well the model describes the system.

12.8.5. Summary

There is no general model that characterizes all field software failure behaviors, but there are methods for determining which model is appropriate and for increased accuracy of the predictions. In many situations simpler models may be quite sufficient For example, the two most frequently used software reliability growth models with field data and usage-time failure intensity metrics appear to be the GO and the MO models. From a practical standpoint, reliability is only defined in a customer's environment; a system that is not used will not fail. Therefore, in addition to understanding faults (which are a function of a developer's error in interpretation, logic, or implementation), the customer's usage of the product is very important to understanding the current or future reliability of a system. If usage data is difficult to obtain (as is often the case), then other models can be used for predicting field failure rates based on calendar time as their time component. However, the type of model used and how it is employed can vary greatly with calendar-based models. There is less consistent empirical information about which calendar-time based models are appropriate than is the case with usage time models. In creating models and fitting data, we must remember that although a model may "look" good and "fit" data well, the important aspect of modeling is not how well it fits current data (interpolation)

but how well the future reliability (or extrapolation) is characterized. It is very critical that we use diagnostics (numeric or graphical) that are appropriate for this goal. Certain diagnostics used for traditional statistical models (mean-squared error and r-squared) may or may not be optimal. Of more relevance are graphical and numeric diagnostics that show the trends and linearizations in the failure intensity curve. Distinguishing one curvilinear function from another (which is often done with cumulative failure curves) creates analysis fraught with peril simply due to our perceptual shortcomings. Therefore, we recommend examining diagnostic graphs that confirm linear relationships between (transformations of) the entities λ, μ, and t. We have described some of the requirements of field software reliability analysis. The methods required for implementation include a sound scientific basis in observation, recording, and analysis. Statistical methods are very useful in characterizing failure behavior of a system as well as potentially predicting the future failure behavior. To this end, practitioners use models for characterizations, and these models usually lead to a deeper understanding of the system reliability. Models also provide parameters that are the key indicators of the system. Understanding the parameters, how they behave, and to what they are related is a fundamental aspect of analyzing the reliability of a system. Thus, creating models based on field data provides a sound framework for our understanding of the complex nature of current products which are increasingly dependent on large amounts of software. This framework can then be related to the software development and delivery process so that potential improvements in the process can be implemented which will enhance the reliability of the system.

Field reliability data is usually not as consistent in quality as lab test data. In testing, the time of the failure and the usage of the system is often recorded with great precision. This is usually not the case with field software reliability data. Moreover, failures are often recorded en masse or even grouped outright due to this lack of precision in usage data. Therefore, our models and methods should be robust with respect to the precision of the data. Certain diagnostics related to time-to-next failure may not be applicable to field data. Also, an environment to support this analysis is critical if the information is to be of use in guiding an organization in process improvement. This implies a database system that allows data examination at a high-level (accessing and understanding information from thousands of records simultaneously) in addition to the lower levels in which they are most commonly

used (examining in detail several variables for one record). Subsequent to data access, an environment using some of the tools that are available greatly expedites the analysis process. There are many software packages available today enabling sophisticated field reliability analysis that were non-existent or were inaccessible in the previous decade. In the long term, this leads to more sophisticated analyses, better diagnostic methods, and more useful results.

In addition to reliability models, there are related concepts which we have examined. Rare event analysis is one area where classical models are either impractical or much more natural variation exists due to the nature of the problem. Some classes of failures may be of special interest, and may be considered more important than others. Usually these are failures that are categorized as having life-threatening, or extremely damaging consequences. In that context, the key is identification of these failure modes, and of the events that lead to these failures. Failures modes that are absolutely unacceptable should not be analyzed using only probabilistic methods since these methods are inherently incapable of assuring the level of reliability that is required for such systems. Some other techniques, such as formal methods, should be used to complement the analyses. However, the special event failures to which one is willing to attach probability of occurrence can be analyzed through the concept of risk, as well as through reliability models that have been explicitly and thoroughly validated in the environment and for the application to which they are being applied.

Another concept that is related to software reliability is software availability. Availability models are functions of both software reliability and the field recovery rate of software-based systems. In practice, it may be reasonable to assume that, past the transient region, system recovery rate is constant, while the failure rate is a decreasing function of the in-service time or calendar time. This may allow use of very simple availability models for description and prediction of the empirical availability behavior of an operational system. However, a complete unavailability model for practical systems needs to incorporate time-dependent parameters, as well as more than one operational state and more than one failure state to account for software and other types of causes and different classes of failure duration.

13. Standards and Handbooks

13.1. Reliability Standards & Handbooks

13.1.1. MIL-HDBK-H 108 Sampling Procedures and Tables for Life and Reliability Testing (Based on Exponential Distribution)

This handbook provides procedures and tables based on the exponential distribution for life and reliability testing. It includes definitions required for the use of the life test sampling plans and procedures; general description of life test sampling plans; life tests terminated upon occurrence of preassigned number of failures; life tests terminated at preassigned time; and sequential life test sampling plans.

13.1.2. MIL-HDBK-189 Reliability Growth Management

This document is designed for both managers and analysts covering everything from simple fundamentals to detailed technical analysis. Included are concepts and principles of reliability growth, advantages of managing reliability growth, and guidelines and procedures used to manage reliability growth. It allows the development of a plan that will aid in developing a final system that meets requirements and lowers the life-cycle cost of the fielded system. The document includes sections on benefits, concepts, engineering analysis, and growth models.

13.1.3. MIL-HDBK-217F Reliability Prediction of Electronic Equipment

The purpose of this handbook is to establish and maintain consistent and uniform methods for estimating the inherent reliability of electronic equipment and systems. It provides a common basis for reliability predictions. This handbook includes two basic methods for reliability prediction of electronic equipment. The first method is the part stress analysis prediction technique, employing complex models using detailed stress analysis information as well as environment, quality applications, maximum ratings, complexity, temperature, construction, and a number of other

application-related factors. The second is a simple method called the parts count reliability prediction technique, using primarily the number of parts of each category with consideration of part quality, environments encountered, and maturity of the production process. The simple method is beneficial in early trade-off studies and situations where the detailed circuit design is unknown. The complex method requires detailed study and analysis which is available when the circuit design has been defined. Samples of each type of calculation are provided.

13.1.4. MIL-HDBK-251 Reliability/Design Thermal Applications

This document details approaches to thermal design; methods for the determination of thermal requirements; selection of cooling methods; natural methods of cooling; thermal design for forced air, liquid-cooled, vaporization, and special (heat pipes) cooling systems. Topics covered are the standard hardware program thermal design, installation requirements, thermal evaluation, improving existing designs, and thermal characteristics of parts. Stress analysis methods are emphasized and specific step by step thermal design procedures are given.

13.1.5. MIL-HDBK-263A Electrostatic Discharge Control Handbook for Protection of Electrical and Electronic Parts, Assemblies and Equipment (Excluding Electrically Initiated Explosive Devices)

This handbook provides guidance for developing, implementing and monitoring an ESD control program for electronic parts, assemblies and equipment in accordance with the requirements of MIL-STD-1686. This document includes definitions, causes and effects (including failure mechanisms), charge sources, list and category of electrostatic-sensitive devices by part type, testing, application information, considerations, and protective networks. The specific guidance provided is supplemented by technical data contained in the appendices. Table I provides a cross-reference listing of MIL-STD-1686 requirements, MIL-HDBK-263 guidance, and MIL-HDBK-263 supplementary technical data.

13.1.6. MIL-HDBK-338 Electronic Reliability Design Handbook

This handbook provides procuring activities and contractors with an understanding of the concepts, principles, and methodologies covering all aspects of electronic systems reliability engineering and cost analysis as they relate to the design, acquisition, and deployment of equipment or systems. Currently a two-volume set, it discusses the entire subject, heavily emphasizing the reasons for the reliability discipline. It includes general information, referenced documents, definitions, reliability theory, component reliability design considerations, application guidelines, specification control during acquisition, logistic support, failure reporting and analysis, reliability and maintainability theory, reliability specification allocation and prediction, reliability engineering design guidelines, reliability data collection and analysis, demonstration and growth, software reliability, systems reliability engineering, production and deployment reliability and maintainability (R&M), and R&M management considerations.

13.1.7. MIL-HDBK-344 Environmental Stress Screening of Electronic Equipment

This handbook provides uniform procedures, methods and techniques for planning, monitoring and controlling the cost effectiveness of ESS programs for electronic equipment. It is intended to support the requirements of MIL-STD-785, Task 301, "Environmental Stress Screening" and/or MIL-STD-781 and Task 401, "Environmental Stress Screening".

13.1.8. MIL-STD-690C Failure Rate Sampling Plans and Procedures

This standard provides procedures for failure rate qualification, sampling plans for establishing and maintaining failure rate levels at selected confidence levels, and lot conformance inspection procedures associated with failure rate testing for the purpose of direct reference in appropriate military electronic parts established reliability (ER) specifications. Figures and tables throughout this standard are based on exponential distribution.

13.1.9. MIL-STD-721C Definition of Terms for Reliability and Maintainability

This standard defines terms and definitions used most frequently in specifying Reliability and Maintainability (R & M). Provides a common definition for the Department of Defense and defense contractors.

13.1.10. MIL-STD-756B Reliability Modeling and Prediction

This standard establishes uniform procedures and ground rules for the generating mission reliability and basic reliability models and predictions for electronic, electrical, electromechanical, mechanical, and ordnance systems and equipments. Model complexity may range from a complete system to the simplest subdivision of a system. It details the methods for determining service use (life cycle), creation of the reliability block diagram, construction of the mathematical model for computing the item reliability. Some simple explanations on the applicability and suitability of the various prediction sources and methods are included.

13.1.11. MIL-HDBK-781 Reliability Test Methods, Plans and Environments for Engineering Development, Qualification and Production

This handbook provides test methods, test plans, and test environmental profiles which can be used in reliability testing during the development, qualification, and production of systems and equipment. This handbook is designed to be used with MIL-STD-781. The test methods, test plans, and environmental profile data are presented in a manner which facilitates their use with the tailorable tasks of MIL-STD-781.

13.1.12. MIL-STD-781D Reliability Design Qualification and Production Acceptance Tests: Exponential/ Distribution

This document covers the requirements and provides details for reliability testing during the development, qualification, and production of systems and equipment with an exponential time-to-failure distribution. It establishes the tailorable requirements for reliability testing performed during integrated test programs specified in MIL-STD-785. Task descriptions for Reliability Development/ Growth

Testing (RD/GT), Reliability Qualification Testing (RQT), Production Reliability Acceptance Tests (PRAT), and Environmental Stress Screening (ESS) are defined. Test time is stated in multiples of the design Mean Time Between Failures (MTBF). Specifying any two of three parameters, i.e., lower test MTBF, upper test MTBF, or their ratio, given the desired decision risks, determines the test plan to be utilized. This standard is applicable to six broad categories of equipment, distinguished according to their field service applications.

13.1.13. MIL-STD-785B Reliability Program for Systems and Equipment, Development and Production

This document provides general requirements and specific tasks for reliability programs. It is used for reliability program planning and includes task descriptions for basic application requirements including sections on program surveillance and control, design and evaluation, development and production testing. An appendix for application guidance for implementation of reliability program requirements is also included. The subsections are in the form of purpose, task description, and details to be specified by the procuring activity. This is a program management document, not a typical detailed what-to-do standard document.

13.1.14. MIL-STD-790E Reliability Assurance Program for Electronic Parts Specifications

This document establishes the criteria for electronic and fiber optic parts product assurance programs which are to be met by manufacturers qualifying electronic parts to specification. Typical topics covered are document submission, organizational structure, test facilities, and failure analysis reports.

13.1.15. MIL-STD-1543B Reliability Program Requirements for Space and Missile Systems

This document establishes uniform reliability program requirements and tasks for use during design, development, fabrication, test, and operation of space and launch vehicles. Topics covered in this document are design for reliability; failure mode, effects, and criticality analysis (FMECA), reliability analysis; modeling and prediction; discrepancy and failure reporting; maximum preacceptance operation; effects of

testing, storage, shelf life; packaging, transportation, handling, and maintainability. It gives application guidance and an appendix for FMEA for space and launch vehicle systems.

13.1.16. MIL-STD-1629A Procedures for Performing a Failure Mode, Effects, and Criticality Analysis

This document shows how to perform a Failure Mode, Effects, and Criticality Analysis (FMECA). It establishes requirements and procedures for performing a FMECA to systematically evaluate and document, by item failure mode analysis, the potential impact of each functional or hardware failure on mission success, personnel and system safety, system performance, maintainability, and maintenance requirements. Each potential failure is ranked by the severity of its effect in order that appropriate corrective actions may be taken to eliminate or control the high risk items. It details the functional block diagram modeling method, defines severity classification and criticality numbers. It provides sample formats for a FMEA, criticality analysis, FMEA and criticality analysis maintainability information sheet, and damage mode and effects analysis sheet. The document also provides several examples.

13.1.17. MIL-STD-1686B Electrostatic Discharge Control Program for Protection of Electrical and Electronic Parts, Assemblies and Equipment (Excluding Electrically Initiated Explosive Devices)

The purpose of this standard is to establish the requirements for an ESD control program to minimize the effects of ESD on parts, assemblies, and equipment. This standard defines the requirements for an ESD control program for electrical and electronic parts, assemblies, and equipment, susceptible to damage from ESD. It covers identification, testing, classification, assembly and equipment design criteria protected work areas, handling procedures, training, marking of documentation and hardware, protective covering, packaging and marking, and installation for assemblies and equipment. Also included are quality assurance requirements, data requirements, audits and reviews. Refer to MIL-HDBK-263 for how-to information.

13.1.18. MIL-STD-2074 Failure Classification for Reliability Testing

This document establishes criteria for classification of failures occurring during reliability testing. This classification into relevant or nonrelevant categories allows the proper generation of MTBF reports. This document applies to any reliability test, including, but not limited to, tests performed in accordance with MIL-STD-781.

13.1.19. MIL-STD-2155 Failure Reporting, Analysis and Corrective Action System (FRACAS)

This document establishes uniform requirements and criteria for a Failure Reporting, Analysis, and Corrective Action System (FRACAS) to implement the FRACAS requirement of MIL-STD-785.

13.1.20. MIL-STD-2164 Environment Stress Screening Process for Electronic Equipment

This document defines the requirements for ESS of electronic equipment, including environmental test conditions, duration of exposure, procedures, equipment operation, actions taken upon detection of defects, and test documentation. The document provides for a uniform ESS to be utilized for effectively disclosing manufacturing defects in electronic equipment.

13.2. Maintainability Standards & Handbooks

13.2.1. MIL-STD-470B Maintainability Program Requirements for Systems and Equipment

This document includes application requirements, tailorable maintainability program tasks, and an appendix with an application matrix and guidance and rationale for task selection. The topics covered are program surveillance and control, design and analysis, modeling, allocations, predictions, failure mode and effects analysis, and maintainability design criteria. Each task item includes a purpose, task description, and details to be specified. Software maintainability is not covered by this document.

13.2.2. MIL-STD-471A Maintainability Verification/ Demonstration/ Evaluation

This document provides procedures and test methods for verification, demonstration, and evaluation of qualitative and quantitative maintainability requirements. It also provides for qualitative assessment of various integrated logistic support factors related to and impacting the achievement of maintainability parameters and item downtime, e.g. technical manuals, personnel, tools and test equipment, maintenance concepts and provisioning.

13.2.3. MIL-HDBK-472 Maintainability Prediction

This document is to familiarize project managers and design engineers with maintainability prediction procedures. It provides the analytic foundation and application details of five prediction methods. Each procedure details applicability, point of application, basic parameters of measure, information required correlation, and cautionary notes. The highlights of each maintainability prediction procedure are presented in a clear and intelligible manner and include useful supplementary information applicable to specific procedures. Maintainability Prediction Procedures I and III are applicable solely to electronic systems and equipments. Procedures II and IV can be used for all systems and equipments. In applying Procedure II to non-electronic equipments the appropriate task times must be estimated. Procedure V can be used to predict maintainability parameters of avionics, ground and shipboard electronics at the organizational, intermediate and depot levels of maintenance.

13.2.4. DOD-HDBK-791 Maintainability Design Techniques

This handbook supplies information on incorporating maintainability into Army materiel design. It defines maintainability and discusses its importance, quantitative measurement, and incorporation into the design process. Other subjects discussed in detail cover simplification, standardization and interchangeability, accessibility, modularization, identification and labeling, testability and diagnostic techniques, preventive maintenance, human factors, and environmental factors as they relate to maintainability.

13.2.5. MIL-STD-1591 On Aircraft, Fault Diagnosis, Subsystems, Analysis/Synthesis of

This document establishes uniform criteria for conducting trade studies to determine the optimal design for an on-aircraft fault diagnosis/isolation system. This document is applicable where a selection can be made between such alternatives as central computer controlled on-board centrally polled built-in test equipment (BITE), decentralized BITE, detached Aerospace Ground Equipment (AGE), etc., or combinations of the preceding. The fault diagnosis/isolation systems of interest are those used to diagnose/isolate faults at the flight line (organizational) level of maintenance. This document also provides a cost model and a maintainability labor power model.

13.2.6. MIL-STD-1843 Reliability-Centered Maintenance for Aircraft, Engines and Equipment

This document, which is based on the Airline/Manufacturer Maintenance Program Planning Document MSG-3, outlines the procedures for developing preventive maintenance requirements through the use of Reliability-Centered Maintenance Analysis (RCMA) for Air Force aircraft and engine systems, aircraft and engine structures and equipment, including peculiar and common Support Equipment (SE) Communications and Electronics (C-E) equipment, vehicles, weapons and other similar equipment items.

13.2.7. MIL-STD-2084 Maintainability of Avionic & Electronic Systems and Equipment

This document covers the common maintainability design requirements to be used in military specifications for avionic and electronic systems and equipment.

13.2.8. MIL-STD-2165A Testability Programs for Electronic Systems & Equipment

This document is intended to prescribe a systematic approach for establishing and conducting a testability program. It describes a uniform approach to testability program planning, establishment of diagnostic concepts and testability (including BIT) requirements, testability and test design and assessment, and requirements for

conducting testability program reviews. Relevant tasks in this document are to be applied during the conceptual phase, demonstration and validation phases, full-scale development phase and production phase of the acquisition process.

13.2.9. DOD-STD-1701 Hardware Diagnostic Test System Requirements

This document establishes the general procedures, terms and conditions governing the preparation and completion of a hardware diagnostic test system.

13.2.10. MIL-STD-2173 Reliability-Centered Maintenance Requirements for Naval Aircraft, Weapons Systems and Support Equipment

This document is used to provide procedures for a Reliability-Centered Maintenance analysis for naval aircraft, weapons systems, and support equipment. This document is used during development of new systems and equipment, and by analysts and auditors within the Naval Air Systems Command for determining preventive maintenance requirements and developing age exploration requirements. The document can also be used to update the initial reliability-centered maintenance analysis and analyze newly discovered failure modes.

13.2.11. MIL-STD-001591A Subsystem Analysis/Synthesis of Command, Control & Communication (C3) System Component Fault Diagnosis

This document establishes uniform criteria for conducting trade studies to determine the optimal design for command, control and communication system and component fault diagnosis/isolation subsystems, These types of systems are referred to as Fault Identification & Test Subsystems (FITS). FITS include the hardware and/or software necessary for the detection and isolation of failures.

13.3. Safety Standards & Handbooks

13.3.1. MIL-HDBK-274 Electrical Grounding for Aircraft Safety

The purpose of this handbook is to provide aircraft maintenance personnel with the information required for electrical safety grounding of each type of operational aircraft in the U.S. Navy inventory. In addition, this handbook provides background information pertaining to the operational concerns for aircraft grounding, static electricity theory and how it affects aircraft, and techniques used for measurement of grounding points.

13.3.2. MIL-HDBK-764 System Safety Engineering Design Guide For Army Materiel

This handbook presents system safety considerations for use in designing army materiel. The areas covered include safety engineering concepts and objectives, system safety analysis, hazard analysis, software analysis, and general design application considerations.

13.3.3. MIL-HDBK-828 Laser Range Safety

The purpose of this handbook is to provide uniform guidance in evaluations for the safe use of military lasers and laser systems on DOD military reservations or military-controlled areas worldwide. This handbook is intended to supplement each military service s normal procedures for laser ranges.

13.3.4. MIL-STD-882C System Safety Program Requirements

This document provides requirements for developing and implementing a System safety program to identify the hazards of a system and to impose design requirements and management controls to prevent mishaps by eliminating hazards or reducing risks. It applies to every activity of the system life cycle; e.g., research, technology development, design, test and evaluation, production, construction, checkout/calibration, operation, maintenance and support, modification and disposal. Twenty-two tasks are defined in the areas of program management and control and design and evaluation. Typical tasks are system safety program plan, preliminary hazard analysis, and software hazard analysis. An appendix is provided to give some rationale and methods for satisfying the requirements previously detailed.

13.3.5. MIL-STD-1247C Markings, Functions and Hazard Designations of Hose, Pipe, and Tube Lines for Aircraft Missiles, and Space Systems

This document is intended for use in the establishment of material labeling requirements for identification, function, sub-function, pressures, hazards and direction of flow for pipes, hoses, and tube lines used in aircraft, missile, space systems, and support equipment. The use of colors, words and symbols to identify the functions of such items (to include approved abbreviations), and the dimensions of labeling items such as tags, tapes, and bands, are specifically prescribed. This document is designed to result in rapid servicing of functional systems to return them to full operation and are an integral part of the complete system.

13.3.6. MIL-STD-1425A Safety Design Requirements for Military Lasers and Associated Support Equipment

This document defines safety design requirements for military laser systems and associated support equipment. These requirements are the minimum requirements necessary to control the hazards caused directly by laser radiation. Associated system hazards, such as electrical shock, toxic chemicals, high pressure, etc., are controlled through the selection of appropriate requirements in other standards and specifications. A comprehensive system safety program in accordance with MIL-STD-882 should be used to identify and control all hazards unique to the specific laser.

13.3.7. MIL-STD-1576 Electroexplosive Subsystem Safety Requirements and Test Methods for Space Systems

The purpose of this document is to insure the safety of personnel, launch site facilities, and space vehicles from the hazards resulting from electroexplosive subsystem inadvertent initiation. The requirements and test methods contained in this document are not intended to insure all electroexplosive subsystem performance requirements except in those cases where failure to perform would create a hazard to personnel, launch site facilities, and space vehicles. The electroexplosive subsystem is composed of all components from the power source to, and including, the

electroexplosive device; safe and arm devices, arm/disarm switches, relays and all electrical wiring used to monitor, control, arm and fire ordnance are specifically included. This Standard applies to all space vehicle systems (e.g., launch vehicles, upper stages, boosters, payloads, and related systems using electroexplosive devices.

13.4. Safety Standards & Handbooks

13.4.1. MIL-HDBK-274 Electrical Grounding for Aircraft Safety

The purpose of this handbook is to provide aircraft maintenance personnel with the information required for electrical safety grounding of each type of operational aircraft in the U.S. Navy inventory. In addition, this handbook provides background information pertaining to the operational concerns for aircraft grounding, static electricity theory and how it affects aircraft, and techniques used for measurement of grounding points.

13.4.2. MIL-HDBK-764 System Safety Engineering Design Guide For Army Materiel

This handbook presents system safety considerations for use in designing army materiel. The areas covered include safety engineering concepts and objectives, system safety analysis, hazard analysis, software analysis, and general design application considerations.

13.4.3. MIL-HDBK-828 Laser Range Safety

The purpose of this handbook is to provide uniform guidance in evaluations for the safe use of military lasers and laser systems on DOD military reservations or military-controlled areas worldwide. This handbook is intended to supplement each military service s normal procedures for laser ranges.

13.4.4. MIL-STD-882C System Safety Program Requirements

This document provides requirements for developing and implementing a System safety program to identify the hazards of a system and to impose design requirements

and management controls to prevent mishaps by eliminating hazards or reducing risks. It applies to every activity of the system life cycle; e.g., research, technology development, design, test and evaluation, production, construction, checkout/calibration, operation, maintenance and support, modification and disposal. Twenty-two tasks are defined in the areas of program management and control and design and evaluation. Typical tasks are system safety program plan, preliminary hazard analysis, and software hazard analysis. An appendix is provided to give some rationale and methods for satisfying the requirements previously detailed.

13.4.5. MIL-STD-1247C Markings, Functions and Hazard Designations of Hose, Pipe, and Tube Lines for Aircraft Missiles, and Space Systems

This document is intended for use in the establishment of material labeling requirements for identification, function, sub-function, pressures, hazards and direction of flow for pipes, hoses, and tube lines used in aircraft, missile, space systems, and support equipment. The use of colors, words and symbols to identify the functions of such items (to include approved abbreviations), and the dimensions of labeling items such as tags, tapes, and bands, are specifically prescribed. This document is designed to result in rapid servicing of functional systems to return them to full operation and are an integral part of the complete system.

13.4.6. MIL-STD-1425A Safety Design Requirements for Military Lasers and Associated Support Equipment

This document defines safety design requirements for military laser systems and associated support equipment. These requirements are the minimum requirements necessary to control the hazards caused directly by laser radiation. Associated system hazards, such as electrical shock, toxic chemicals, high pressure, etc., are controlled through the selection of appropriate requirements in other standards and specifications. A comprehensive system safety program in accordance with MIL-STD-882 should be used to identify and control all hazards unique to the specific laser.

13.4.7. MIL-STD-1576 Electroexplosive Subsystem Safety Requirements and Test Methods for Space Systems

The purpose of this document is to insure the safety of personnel, launch site facilities, and space vehicles from the hazards resulting from electroexplosive subsystem inadvertent initiation. The requirements and test methods contained in this document are not intended to insure all electroexplosive subsystem performance requirements except in those cases where failure to perform would create a hazard to personnel, launch site facilities, and space vehicles. The electroexplosive subsystem is composed of all components from the power source to, and including, the electroexplosive device; safe and arm devices, arm/disarm switches, relays and all electrical wiring used to monitor, control, arm and fire ordnance are specifically included. This Standard applies to all space vehicle systems (e.g., launch vehicles, upper stages, boosters, payloads, and related systems using electroexplosive devices.

13.5. Other Relevant Military Documents

13.5.1. MIL-STD-105E Sampling Procedures and Tables for Inspection by Attributes

This document addresses the subjects of sampling plans; lot size; inspection levels; average quality levels (AQLs); classification of defects; multiple sampling; and normal, tightened, and reduced sampling. For equipments where the sequential method of testing, based on operating time, may not be appropriate, this document, based on the success ratio, can be used. It includes numerous tables showing accept-reject levels and operating characteristic curves for sampling plans. The sampling plans described in this document are applicable to AQL's of .01 percent or higher and are therefore not suitable for applications where quality levels in the defective parts per million range can be realized.

13.5.2. MIL-STD-337 Design To Cost

This document prescribes the Design to Cost program objectives and requirements for design and development of systems, subsystems, equipments, and software. It provides general and specific requirements to ensure effective control of the design-

related production and ownership costs. This is accomplished by a Design to Cost program which is planned, documented, implemented, and reviewed in consonance with related technical and management disciplines. Included are requirements for making Life Cycle Cost elements inherent in the critical functional areas of reliability, logistics, and optimization by using tradeoff studies, cost estimation and tracking in the life cycle management acquisition process; requirements for information sharing between government and industry of data and studies relative to the acquisition and ownership costs of the system; requirements for relating Design to Cost to the supportability considerations of the deployed system, to logistic support analyses, and to reliability and maintainability studies.

13.5.3. MlL-STD-454N Standard General Requirements for Electronic Equipment

This document covers the common requirements to be used in military specifications for electronic equipment. It establishes the technical baseline for design and construction of electronic equipment for the Department of Defense. It addresses 75 requirements such as brazing, substitutability, reliability, resistors, and casting. It provides numerous references on the subjects addressed.

This standard was superseded by MIL-STD-2036 several years ago.

13.5.4. MIL-HDBK-728 NonDestructive Testing (NDT)

The handbook is provided as a guide and describes general principles, procedures and safety items, of eddy current, liquid penetrate, magnetic particle, radiographic and ultrasonic testing. This handbook is not a training manual. Nor can it replace other written directives, procedures or specifications. However, it can serve as a ready reference to the important principles and facts relating to the employment of nondestructive testing, inspection and evaluation.

13.5.5. MIL-HDBK-729 Corrosion and Corrosion Prevention Metals

This handbook contains technical information and data pertaining to the corrosion and corrosion protection of metals and alloys. The factors influencing corrosion, forms of corrosive attack including stress-corrosion cracking and hydrogen-stress

cracking (hydrogen embrittlement), corrosion characteristics of metals, and general methods of protecting metals and alloys from corrosion, are covered in condensed form, along with information on corrosion testing. Emphasis is placed upon the type of corrosion encountered by military equipment.

13.5.6. MIL-HDBK-772 Military Packaging Engineering

This handbook covers fundamental principles and practices of military packaging engineering. It provides information concerning materials, the basic causes of deterioration, methods of preservation, and types of preservatives. Also included is information on natural and transportation environments, cost and human engineering factors, and other special military packaging considerations.

13.5.7. MIL-HDBK-798 System Engineer's Design for Discard Handbook

This handbook provides design guidance as well as general information on applicable concepts, techniques, and procedures for practical implementation of a design for discard program. The handbook explains what design for discard means; why design for discard should be implemented; what the design for discard effort should involve; how to implement design for discard in a project; the tradeoffs involved during design; the interfaces with other system disciplines; and the techniques used to evaluate the results of design for discard.

13.5.8. MIL-STD-810E Environmental Test Methods and Engineering Guidelines

The purpose of this document is to standardize the design and conduct of tests for assessing the ability of military equipment to withstand environmental stresses which it will encounter during its life cycle, and to insure that plans and test results are adequately documented. This document provides guidelines for conducting environmental engineering tasks and provides test methods for determining the effects of natural and induced environments on equipment used in military applications. Included in the numerous types of tests detailed are purpose, environmental effects, guidelines for determining test procedures and test conditions, references, apparatus, preparation for test, procedures, information to be recorded.

13.5.9. MIL-STD-883D Test Methods and Procedures for Microelectronics

This document establishes uniform methods, controls, and procedures for designing, testing, identifying and certifying microelectronic devices suitable for use within military and aerospace electronic systems including basic environmental tests to determine resistance to deleterious effects of natural elements and conditions surrounding military and space operations; physical and electrical tests; design, package and material constraints; general marking requirements; workmanship and training procedures; and such other controls and constraints as have been deemed necessary to ensure a uniform level of quality and reliability suitable to the intended applications of those devices. Also covered are test procedures for failure analysis, limit testing, wafer lot acceptance, and destructive physical analysis.

13.5.10. MIL-STD-965B Parts Control Program

The purpose of this document is to establish the guidelines and requirements for implementation of a parts control program. This document describes two procedures covering the submission, review, and approval of program parts selection lists (PPSL). Typical topics covered are PPSL approval, meetings, parts control board, and Military Parts Control Advisory Group.

13.5.11. MIL-STD-975M NASA Standard Electrical, Electronic, and Electro Mechanical (EEE) Parts List

This purpose of this document is to provide equipment designers and manufacturers with electronic parts having quality levels considered to be most acceptable for flight and mission-essential ground support equipment. It provides a means of selecting, procuring, and applying Electrical, Electronic, and Electromechanical (EEE) Parts for use in flight and mission-essential ground support equipment. Three levels of quality are used in this standard. Grade 1 parts are very low risk, higher quality and reliability parts intended for critical applications. Grade 2 parts are low risk, high quality and reliability parts for use in applications not requiring Grade 1 parts. Grade 3 parts are higher risk, good quality and reliability parts but are not recommended for applications requiring high product assurance levels.

13.5.12. MIL-STD-1369 Integrated Logistic Support Program Requirements

This document establishes the requirements, procedures and formats to be used for the development and documentation of an Integrated Logistic Support Program. Integrated Logistic Support, including maintainability requirements, is the result of a planning process designed to aid in achieving maximum overall system effectiveness by taking advantage of the direct relationship which exists between hardware design characteristics and resultant logistic support requirements by considering both, concurrently throughout the design process, in a quantitative manner to provide a basis for weapons/weapon systems/equipment optimization through sound engineering trade-off. Documentation requirements will provide accurate records to assure continuation of an adequate logistics support base throughout the operational life cycle of a weapons system or equipment.

13.5.13. MIL-STD-1388-1A Logistics Support Analysis

This document details Logistic Support Analysis guidelines and requirements. Tasks detail the purpose, task description, task input, and task output. Typical tasks are program planning and control; development of early Logistic Support Analysis strategy; planning; program and design reviews; mission hardware, software, and support; system standardization; early fielding analysis; and supportability assessment.

13.5.14. MIL-STD-1388-2B DOD Requirements for a Logistic Support Analysis Record

This document describes the data element definitions, data field lengths, and formats for Logistic Support Analysis Record (LSAR) data. It identifies the LSAR reports that are generated from the LSAR data and identifies the LSAR relational tables and automated data processing specifications for transmittal and delivery of automated LSAR data.

13.5.15. MIL-STD-1556B Government/Industry Data Exchange Program (GIDEP)

This document defines the requirements for participation in the GIDEP program, which includes the engineering, failure experience, reliability-maintainability, and metrology data interchanges. It is intended to be applied to prime contractors and major subcontractors (who are users of parts) for the government. The reliability-maintainability data interchange contains failure rate and mode and replacement rate data on parts, components, assemblies, subsystems, and materials based on field performance information and reliability test of equipment, subsystems, and systems. This data interchange also contains reports on theories, methods, techniques, and procedures related to reliability and maintainability practices.

13.5.16. MIL-STD-1568B Materials and Processes for Corrosion Prevention and Control in Aerospace Weapons Systems

This document establishes the requirements for materials, processes and techniques, and identifies the tasks required to implement an effective corrosion prevention and control program during the conceptual, validation, development and production phases of aerospace system. The intent is to minimize life cycle cost due to corrosion and to obtain improved reliability.

13.5.17. RAC NPRD Nonelectronic Parts Reliability Data, 1991

This document provides failure rate and failure mode information for mechanical, electromechanical, electrical, pneumatic, hydraulic, and rotating parts. The assumption that the failures of nonelectronic parts follow the exponential distribution has been made because of the virtual absence of data containing individual times or cycles to failure. Generic failure rate tables include environment; application (military or commercial); failure rate; number of records; number failed; and operating hours. A 60 percent confidence interval is used.

13.6. Non-Military Documents

AAMI *HE*
Human Factors Engineering Guidelines and Preferred Practices for the Design of Medical Devices (Second Edition)

ANSI *R15.05-3*
Industrial Robots and Robot Systems - Reliability Acceptance Testing - Guidelines

ANSI/AIAA *G-035*
Human Performance Measurements

ANSI/AIAA *R-013*
Recommended Practice for Software Reliability

ANSI/ASA *S3.18*
Evaluation of Human Exposure to Whole- Body Vibration, Guide for (ASA 38) (R 1993)

ANSI/ASHRAE *55*
Thermal Environmental Conditions for Human Occupancy

ANSI/ESD *S5.1*
REVISED - Human Body Model (HBM) Electrostatic Discharge Sensitivity Testing

ANSI/HFES *100*
Human Factors Engineering of Visual Display Terminal Workstations

ANSI/IEEE *1023*
Guide for the Application of Human Factors Engineering to Systems, Equipment, and Facilities of Nuclear Power Generating Stations

ANSI/IEEE 500 **Guide to the Collection and Presentation of Electrical, Electronic, Sensing Component, and Mechanical Equipment Reliability Data for Nuclear-Power Generating Stations (R 1991)**

ANSI/IEEE *577*
Standard Requirements for Reliability Analysis in the Design and Operation of Safety Systems for Nuclear Power Generating Stations (R 1992)

ANSI/IEEE *762*
Standard Definitions for Use in Reporting Electric Generating Unit Reliability, Availability, and Productivity (R 1993)

ANSI/SAE AIR 4276
Survey Results: Computerization of Reliability, Maintainability & Supportability (RM&S) in Design

---

ANSI/SAE ARP 4032
Human Engineering Considerations in the Application of Color to Electronic Aircraft Displays

---

ANSI/SAE ARP 4107
Aerospace Glossary for Human Factors Engineers

ANSI/SAE J 833
Human Physical Dimensions, Recommended Practice; May 1989

ARINC 628 ITEM 12.0
Reliability

ASA S3.18
Evaluation of Human Exposure to Whole Body Vibration, Guide for (ASA 38) (R 1993)

ASA S3.34
Guide for the Measurement and Evaluation of Human Exposure to Vibration Transmitted to the Hand (ASA 67)

ASAE EP456
Test and Reliability Guidelines (R 1991)

ASHRAE 55
Thermal Environmental Conditions for Human Occupancy

BSI BS 5760: PART 0
1986 Reliability of Constructed or Manufactured Products, Systems Equipment and Components Part 0: Introductory Guide to Reliability

---

BSI BS 5760: PART 1
1985 Reliability of Systems, Equipments and Components Part 1: Guide to Reliability and Maintainability Programme Management

---

BSI BS 5760: PART 2

1994 Reliability of Systems, Equipments and Components Part 2: Guide to the Assessment of Reliability (G)

BSI BS 5760: PART 3
1982 Reliability of Systems, Equipments and Components Part 3: Guide to Reliability Practices: Examples

---

BSI BS 5760: PART 4
1986 Reliability of Systems, Equipments and Components Part 4: Guide to

Specification Clauses Relating to the Achievement and Development of Reliability in New and Existing Items

---

BSI BS 5760: PART 5
1991 Reliability of Systems, Equipment and Components Part 5: Guide to Failure Modes, Effects and Criticality Analysis (FMEA and FMECA) (G)

BSI BS 5760: PART 6
1991 Reliability of Systems, Equipment and Components Part 6: Guide to Programs for Reliability Growth (IEC 1014: 1989) (G)

BSI BS 5760: PART 7
1991 Reliability of Systems, Equipment and Components Part 7: Guide to Fault Tree Analysis (G) (IEC 1025: 1990)

BSI BS 5760: PART 9
1992 Amd 1 Reliability of Systems, Equipment and Components Part 9: Guide to the Block Diagram Technique (AMD 8152) April 15, 1994 (IEC 1078: 1991) (G)

BSI BS 5760: SEC 10
1993 Reliability of Systems, Equipment and Components Part 10: Guide to Reliability Testing Section, General Requirements, Compliance Test Procedures for Steady- State Availability, Guide to Reliability Testing, Compliance Test Plans for Success Ratio (IEC 1025: 1979)

BSI BS 5760: PART 11
1994 Reliability of Systems, Equipment and Components Part 11: Collection of Reliability, Availability, Maintainability and Maintenance Support Data from the Field (IEC 300- 3-2: 1993) (G)

BSI BS 5760: PART 12
1993 Reliability of Systems, Equipment and Components Part 12: Guide to the Presentation of Reliability, Maintainability and Availability Predictions (IEC 863: 1986) (G)

BSI BS 5760: SEC 13.1
1993 Reliability of Systems, Equipment and Components Part 13: Guide to Reliability Test Conditions for Consumer Equipment Section 13.1: Conditions Providing a Low Degree of Simulation for Indoor Portable Equipment (IEC 605-3-1: 1986) (G)

BSI BS 5760: SEC 13.2
1993 Reliability of Systems, Equipment and Components Part 13: Guide to Reliability Test Conditions for Consumer Equipment Section 13.2: Conditions Providing a High Degree of Simulation for Equipment Use in Partially Weatherprotected Locations

---

BSI BS 5760: SEC 13.3
1993 Reliability of Systems, Equipment and Components Part 13: Guide to Reliability Test Conditions for Consumer Equipment Section 13.3: Conditions

Providing a Low Degree of Simulation for Equipment for Stationary Use in Partially Weatherprotected Locations (IEC 605-3- 3: 1992) (G)

BSI BS 5760: SEC 13.4
1993 Reliability of Systems, Equipment and Components Part 13: Reliability Test Conditions for Consumer Equip. Section 13.4:Conditions Providing a Low Degree of Simulation for Equipment for Portable and Non-Stationary Use (IEC 605-3-4: 1992) (G)

BSI BS 5760: PART 14
1993 Reliability of Systems, Equipment and Components Part 14: Guide to Formal Design Review (IEC 1160: 1992) (G)

BSI BS EN 61078
1994 Amd 1 Reliability of Systems, Equipment and Components Part 9: Guide to the Block Diagram Technique (AMD 8152) April 15, 1994 (IEC 1078: 1991) (G)

BSI HANDBOOK NO.22 PART 2
1992 Reliability and Maintainability (G)

CAA CHAPTER B7-1 APP 08.83
General Reliability

CAA CHAPTER J2-1 APP#3 09.66
System Reliability

CAA CHAPTER K6-12 APP1 10.92
Systems Reliability

CECC CECC 00 801 ISSUE 1 **Preliminary Guidance Document: Pi-Q Factors of CECC Approved Components for Use in Reliability Predictions (En, Fr, Ge) AMD 1 (En, Fr, Ge)**

CECC CECC 00 804 ISSUE 1
Guidance Document: Interpretation of "EN 29000" - Reliability Aspects for Electronic Components (En, Fr, Ge)

CENELEC EN 61078
Analysis Techniques for Dependability - Reliability Block Diagram Method (IEC 1078 : 1991)

CENELEC ETR 039
Human Factors (HF); Human Factors Standards for Telecommunications Applications

---

CENELEC ETR 051
Human Factors (HF); Usability Checklist for Telephones Basic Requirements

---

CENELEC ETR 070
Human Factors (HF); the Multiple Index Approach (MIA) for the Evaluation of Pictograms

CENELEC ETR 095
Human Factors (HF); Guide for Usability Evaluations of Telecommunications Systems and Services

CENELEC ETR 096
Human Factors (HF); Phone Based Interfaces (PBI) Human Factors Guidelines for the Design of Minimum Phone Based User Interface to Computer Services

CENELEC ETR 113
Human Factors (HF); Results of an Evaluation Study of Pictograms for Point-to-Point Videotelephony

CENELEC HD 485
Analysis Techniques for System Reliability - Procedure for Failure Mode and Effects Analysis (FMEA)

CENELEC HD 485 S1
Analysis Techniques for System Reliability - Procedure for Failure Mode and Effects Analysis (FMEA)

CEPT T/N 45-01 E
Testing the Compliance of an Equipment with Its Reliability, Maintainability and Availability Specifications

CEPT T/SF 40 E
Human Factor Aspects of Visual Display Terminals for Telecommunication Services

CNS B8006
Glossary of Terms for Reliability (General) (Oct)(11381)

CNS C5029
General Rules for Reliability Assured Electronic Components (Jan)(4901)

CNS C5155
Data Processing Vocabulary (Part 14: Reliability, Maintenance and Availability) (Aug)(10242)

CNS C6303
Method of Test for Reliability of Household Audio Product (Oct)(12120)

CNS C6304 **Method of Test for Reliability of Video Product (Oct)(12121)**

CSA CAN/CSA- Q632-90
Reliability and Maintainability Management Guidelines; (Gen Instr 1)

CSA CAN/CSA- Q633-90
Reliability, Availability, and Maintainability Design Guide for Electronic Products; (Gen Instr 1)

CSA Q636-93
Guidelines and Requirements for Reliability Analysis Methods; (Gen Instr 1)

EIA JEP70
Quality and Reliability Standards

EIA RB4-A
Reliability Quantification

EIA RB9
Failure Mode and Effect Analyses

ESD S5.1
REVISED - Human Body Model (HBM) Electrostatic Discharge Sensitivity Testing

---

ETSI ETR 039
Human Factors (HF); Human Factors Standards for Telecommunications Applications

---

ETSI ETR 051
Human Factors (HF); Usability Checklist for Telephones Basic Requirements

---

ETSI ETR 068
Human Factors (HF); European Standardisation Situation of Telecommunications Facilities for People with Special Needs

---

ETSI ETR 070
Human Factors (HF); the Multiple Index Approach (MIA) for the Evaluation of Pictograms

---

ETSI ETR 095
Human Factors (HF); Guide for Usability Evaluations of Telecommunications Systems and Services

---

ETSI ETR 096
Human Factors (HF); Phone Based Interfaces (PBI) Human Factors Guidelines for the Design of Minimum Phone Based User Interface to Computer Services

---

ETSI ETR 113
Human Factors (HF); Results of an Evaluation Study of Pictograms for Point-to-Point Videotelephony

---

ETSI ETS 300 295
Human Factors (HF); Specification of Characteristics of Telephone Services Tones When Locally Generated in Terminals

ETSI PRETS 300 295
Human Factors (HF); Specification of Characteristics of Telephone Services Tones When Locally Generated inTerminals

ETSI PRETS 300 375
Human Factors (HF); Pictograms for Point- to-Point Videotelephony

EURO DEC/81/11953
Reliability Military Data Exchange guide (2nd Edition)

EURO EC/EEPSG/73/1944
Standard Format for Presentation of Reliability and Maintainability Information for Equipment Suppliers to Prime Constructors

EURO PSC/83/12418
Supply of Basic Maintainability and Reliability Data

I.E.S.
Glossary of Reliability Growth Terms

I.E.S.
Reliability, Design, Test, and Evaluation Publications

IEC 1014
Programmes for Reliability Growth First Edition

IEC 1078
Analysis Techniques for Dependability - Reliability Block Diagram Method First Edition (CENELEC EN 61078: 1993)

IEC 1123
Reliability Testing Compliance Test Plans for Success Ratio First Edition

IEC 300
Reliability and Maintainability Management Second Edition

IEC 362
Guide for the Collection of Reliability, Availability, and Maintainability Data from Field Performance of Electronic Items First Edition

IEC 605
Equipment Reliability Testing

IEC 706 PT 1
Guide on Maintainability of Equipment Part 1: Sections One, Two and Three Introduction, Requirements and Maintainability Programme First Edition

IEC 706 PT 2
Guide on Maintainability of Equipment Part 2: Section Five - Maintainability Studies During the Design Phase First Edition

IEC 706 PT 3
Guide on Maintainability of Equipment Part 3: Sections Six and Seven Verification and Collection, Analysis and Presentation of Data First Edition

IEC 706 PT 4
Guide on Maintainability of Equipment Part 4: Section 8 - Maintenance and Maintenance Support Planning First Edition

IEC 812
Analysis Techniques for System Reliability - Procedure for Failure Mode and Effects Analysis (FMEA) First Edition

IEC 863
Presentation of Reliability, Maintainability and Availability Predictions First Edition

IEEE 352
Guide for General Principles of Reliability Analysis of Nuclear Power Generating Station Safety Systems (R 1994) (ANSI/IEEE)

IEEE 500
Guide to the Collection and Presentation of Electrical, Electronic, Sensing Component, and Mechanical Equipment Reliability Data for Nuclear-Power Generating Stations (R 1991) (ANSI/IEEE)

IEEE 500 P&V
Standard Reliability Data for Pumps and Drivers, Valve Actuators, and Valves

IEEE 577
Standard Requirements for Reliability Analysis in the Design and Operation of Safety Systems for Nuclear Power Generating Stations (R 1992) (ANSI/IEEE)

IEEE 762
Standard Definitions for Use in Reporting Electric Generating Unit Reliability, Availability, and Productivity (R 1993) (ANSI/IEEE)

IPC D-330 1.2.10
MIL-STD-781 (Reliability Design Qualification and Production Acceptance Tests: Exponential Distribution); Revision C - October 1977, Notice 1 - March 1981 (Design Guide)

IPC D-330 1.2.11
MIL-STD-790 (Reliability Assurance Program for Electronic Parts Specifications); Revision D - May 1986 (Design Guide)

IPC D-330 1.2.18
DOD-STD-2000-1 (Soldering Technology, High Quality/High Reliability); Revision A - March 1985 (Design Guide)

IPC *D-330* *1.2.19*
DOD-STD-2000-2 (Part and Component Mounting for High Quality/High Reliability Soldered Electrical and Electronic Assemblies) (Design Guide)

IPC *D-330* *1.2.20*
DOD-STD-2000-3 (Criteria for High Quality/High Reliability Soldering Technology) (Design Guide)

IPC *D-330* *1.3.15*
MIL-S-45743 (Soldering, Manual Typed, High Reliability Electrical and Electronic Equipment); Revision E - October 1976, Int. Amendment 4 - September 1982 (Design Guide)

IPC *D-330* *2.3.4.1*
Reliability (Design Guide)

ISO *8107*
Nuclear Power Plants - Maintainability - Terminology First Edition

---

JIS *C* *5700*
General Rules for Reliability Assured Electronic Components (R 1982)

JIS *X* *0014*
Glossary of Terms Used in Information Processing (Reliability, Maintenance and Availability)

JIS *Z* *8500*
Ergonomics - Anthropometric and Biomechanic Measurements

MOD *UK* *DSTAN* *00-25:PART* *1*
Human Factors for Designers of Equipment Part 1: Introduction Issue 2 (09.87); Amendment 1 Corrections

---

MOD *UK* *DSTAN* *00-25:PART* *2*
Human Factors for Designers of Equipment Part 2: Body Size Issue 1 (08.85)

MOD *UK* *DSTAN* *00-25:PART* *3*
Human Factors for Designers of Equipment Part 3: Body Strength and Stamina Issue 1 (04.84); Corrigendum

---

MOD *UK* *DSTAN* *00-25:PART* *4*
Human Factors for Designers of Equipment Part 4: Workplace Design Interim Issue 1 (08.91)

MOD *UK* *DSTAN* *00-25:PART* *5*
Human Factors for Designers of Equipment Part 5: Stresses and Hazards Interim Issue 1 (05.92)

MOD *UK* *DSTAN* *00-25:PART* *6*
Human Factors for Designers of Equipment Part 6: Vision and Lighting Issue 1 (08.86)

MOD UK DSTAN 00-25:PART 7
Human Factors for Designers of Equipment Part 7: Visual Displays Issue 1 (10.86)

MOD UK DSTAN 00-25:PART 8
Human Factors for Designers of Equipment Part 8: Auditory Information Interim Issue 1 (04.89)

MOD UK DSTAN 00-25:PART 9
Human Factors for Designers of Equipment Part 9: Voice Communication Interim Issue 1 (04.91)

MOD UK DSTAN 00-25:PART 10
Human Factors for Designers of Equipment Part 10: Controls Interim Issue 1 (12.92)

MOD UK DSTAN 00-25:PART 11
Human Factors for Designers of Equipment Part 11: Design for Maintainability Issue 1 (08.88)

MOD UK DSTAN 00-25:PART 12
Human Factors for Designers of Equipment Part 12: Systems Issue 1 (07.89)

MOD UK DSTAN 00-40:PART 1
Reliability and Maintainability Part 1: Management Responsibilities and Requirements for Programmes and Plans Issue 2 (07.87) (ARMP-1)

MOD UK DSTAN 00-40:PART 1
Reliability and Maintainability Part 1: Management Responsibilities and Requirements for Programmes and Plans Issue 2 (07.87) (ARMP-1)

MOD UK DSTAN 00-40:PART 2
Reliability and Maintainability Part 2: General Application Guidance on the Use of Part 1 (ARMP-1) Issue 1 (06.88) (ARMP-2)

MOD UK DSTAN 00-40:PART 3
MoD Practices and Procedures for Reliability and Maintainability Part 3: Application of National R and M Documents Issue 1 (08.89) (ARMP-3)

MOD UK DSTAN 00-40:PART 4
Reliability and Maintainability Part 4: Guidance for Writing NATO R & M Requirements Documents Issue 1 (09.91) (ARMP-4)

MOD UK DSTAN 00-40:PART 5
Reliability and Maintainability Part 5: Guidance on R & M Training Issue 1 (02.89) (ARMP-5)

MOD UK DSTAN 00-40:PART 6
Reliability and Maintainability Part 6: In-Service R & M Issue 1 (12.88) (ARMP-6)

MOD UK DSTAN 00-40:PART 8
Reliability and Maintainability Part 8: Procurement of Off - The - Shelf Equipment (ARMP - 8) Issue 1 (07.92)

MOD UK DSTAN 00-41
Reliability and Maintainability MoD Guide to Practices and Procedures Issue 3 (06.93) (Supersedes All Previously Issued Separate Parts)

MOD UK DSTAN 00-43:PART 1
Reliability and Maintainability Assurance Activity Part 1: In-Service Reliability Demonstrations Issue 1 (01.93)

MOD UK DSTAN 00-44:PART 1
Reliability and Maintainability Data Collection and Classification Part 1: Maintenance Data & Defect Reporting in the Royal Navy, the Army and the Royal Air Force Issue 1 (03.93) (Supersedes Def Stan 05-59)

MOD UK DSTAN 00-44:PART 2
Reliability and Maintainability Data Collection and Classification Part 2: Data Classification and Incident Sentencing - General Issue 1 (04.94)

MOD UK DSTAN 00-5: PART 1
Design Criteria for Reliability, Maintainability and Maintenance of Land Service Materiel Part 1: General Requirements Issue 3 (02.79); Amendment 1

MOD UK DSTAN 00-5: PART 2
Design Criteria for Reliability, Maintainability and Maintenance of Land Service Materiel Part 2: Mechanical Aspects Issue 3 (02.79)

MOD UK DSTAN 00-5: PART 3
Design Criteria for Reliability, Maintainability and Maintenance of Land Service Materiel Part 3: Electrical and Electronic Aspects Issue 3 (02.79)

MOD UK DSTAN 00-5: PART 4
Design Criteria for Reliability, Maintainability and Maintenance of Land Service Materiel Part 4: Optical Aspects Issue 1 (02.79)

MOD UK DSTAN 05-48
Reliability of a Series System Issue 1 (03.78)

MOD UK DSTAN 05-63
Guidelines for Classifying Incidents for Reliability Estimation of Tracked and Wheeled Vehicles Issue 1 (10.84)

MOD UK NES 1017
Requirements for Maintainability Demonstrations of Naval Systems Issue 3 (01.93)

NATO ARMP-1 AMD 0
NATO Requirements for Reliability and Maintainability

NATO ARMP-1 ED 2
NATO Requirements for Reliability and Maintainability

NATO ARMP-2 AMD 0
General Application Guidance on the Use of ARMP-1

NATO ARMP-2 ED 2
General Application Guidance on the Use of ARMP-1

NATO ARMP-3 AMD 0
Application of National R and M Documents

NATO ARMP-4 ED 1 AMD 0
Guidance for Writing NATO R & M Requirements Documents

NATO ARMP-5 AMD 0
Guidance on Reliability and Maintainability Training

NATO ARMP-6 AMD 0
In-Service R & M

NATO ARMP-8 ED 1 AMD 0
Reliability & Maintainability in the Procurement of Off-The- Shelf Equipment

NATO STANAG 4174 ED 1 AMD 1
Allied Reliability and Maintainability Publications

NFPA FPEH SEC 4-5
Reliability (SFPE Handbook of Fire Protection Engineering, 1st Ed.)

RAC CRTA-FMECA
Failure Mode, Effects, and Criticality Analysis (FMECA)

RAC EEMD-1
Electronic Equipment Maintainability Data

RAC EERD-2
Electronic Equipment Reliability Data

RAC MDR-21
Microcircuit Device Reliability Trend Analysis

RAC MFAT-2
Characterization and Failure Analysis Techniques a Procedural Guide

RAC NONOP-1
Nonoperating Reliability Databook

RAC NPRD
Nonelectronic Parts Reliability Data

RAC *NPS*
Mechanical Applications in Reliability Engineering

RAC *NPS-1*
Analysis Techniques for Mechanical Reliability

RAC *PRIM*
Primer for DOD Reliability, Maintainability, Safety, and Logistics Standards

RAC *RDSC-1*
Reliability Sourcebook

RAC *RMST-93*
Reliability & Maintainability Software Tools

RAC *SOAR-2*
Practical Statistical Analysis for the Reliability Engineer

RAC *SOAR-3*
IC Quality Grades: Impact on System Reliability and Life Cycle Cost

RAC *SOAR-4*
Confidence Bounds for System Reliability

RAC *SOAR-5*
Surface Mount Technology: A Reliability Review

RAC *SOAR-6*
ESD Control in the Manufacturing Environment

RAC *SOAR-8*
Process Action Team Handbook

RAC *TOOLKIT*
Reliability Engineer's Toolkit

SAA *AS* *2529*
Collection of Reliability, Availability and Maintainability Data for Electronics and Similar Engineering Use (R 1994)

SAA *AS* *2530*
Presentation of Reliability Data on Electronic and Similar Components (R 1994)

SAA *AS* *3930*
Reliability and Maintainability - Introductory Guide (in Professional Package 46)

SAA *AS* *3960*
Guide to Reliability and Maintainability Program Management

SAE *AE-9*
Automotive Electronics Reliability Handbook; February 1987

SAE *AIR* *4276*
Survey Results: Computerization of Reliability, Maintainability & Supportability (RM&S) in Design

SAE *ARD* *50010*
Recommended RMS Terms and Parameters

SAE *ARD* *50046*
RMS Information Sourcebook

SAE *ARP* *4107*
Aerospace Glossary for Human Factors Engineers

SAE *ARP* *4153*
Human Interface Criteria for Collision

SAE *ARP* *4155*
Human Interface Design Methodology for

SAE *ARP* *813A*
Maintainability Recommendations for Aircraft Wheels and Brakes (R 1988)

SAE *J* *1213/2*
Glossary of Reliability Terminology Associated with Automotive Electronics, Information Report; October 1988

SAE *J* *1460*
Human Mechanical Response Characteristics, Information Report; March 1985

SAE *J* *833*
Human Physical Dimensions, Recommended

SAE *J* *885*
Human Tolerance to Impact Conditions as Related to Motor Vehicle Design, Information Report; July 1986

SEMI *E10*
Guideline for Definition and Measurement of Equipment Reliability, Availability, and Maintainability (RAM)

APPENDIX A - Field Data

Dataset 1

Field data from a larger release of a telecommunications switch software. **Data** is calendar-time, **%Cum SW Failures** is the percentage of the total number of software failure experienced in the calendar interval reported in the table, **%Cum Usage Time** is the percentage of the total in-service time accumulated over the calendar interval reported, and **%Sites** is the percentage of sites that have this version of the software release loaded on a given date. Note that the data has been normalized to protect proprietary information.

Date	%Cum SW Failures	%Cum Usage Time	%Sites	Date	%Cum SW Failures	%Cum Usage Time	%Sites
0	0	0.0	2	70	77	47.9	99
1	1	0.1	2	71	79	49.0	98
2	1	0.1	2	72	79	50.2	97
3	1	0.1	2	73	80	51.3	95
4	1	0.1	2	74	81	52.4	95
5	1	0.2	2	75	82	53.4	93
6	2	0.2	3	76	83	54.5	91
7	2	0.2	3	77	83	55.5	90
8	2	0.3	4	78	84	56.6	89
9	2	0.3	5	79	85	57.6	88
10	2	0.4	5	80	85	58.6	85
11	3	0.4	6	81	86	59.6	82
12	4	0.5	7	82	86	60.5	80
13	4	0.6	7	83	87	61.4	78
14	5	0.7	9	84	87	62.3	78
15	7	0.8	13	85	88	63.2	76
16	7	1.0	16	86	88	64.1	75
17	8	1.1	16	87	89	64.9	74
18	9	1.5	17	88	89	65.8	72
19	10	1.7	19	89	90	66.6	71
20	10	2.0	22	90	90	67.4	71
21	11	2.2	25	91	90	68.2	69
22	13	2.6	29	92	91	69.0	68
23	14	2.9	33	93	91	69.8	67
24	16	3.3	37	94	92	70.5	65
25	17	3.8	41	95	92	71.3	64
26	18	4.3	46	96	92	72.0	63
27	20	4.8	49	97	92	72.7	62
28	22	5.4	54	98	93	73.4	61
29	25	6.1	56	99	93	74.1	61
30	26	6.7	60	100	93	74.8	60
31	28	7.4	63	101	94	75.5	59
32	30	8.2	65	102	94	76.2	59