FOREWORD

This notebook provides a reliability assurance methodology to predict and estimate the reliability of systems that employ both hardware and software subsystems. Since the methods used to predict the reliability of hardware systems are well established, this notebook concentrates on the methods to predict and estimate the reliability of software configuration items and methods for combining hardware and software reliability metrics into an overall system parameter.
# Table of Contents

1.0 INTRODUCTION 1-1

2.0 APPLICABLE DOCUMENTS 2-1

3.0 DEFINITIONS AND SYMBOLS 3-1
   3.1 Definitions of Terms 3-1
   3.2 Abbreviations 3-4
   3.3 Mathematical Symbols 3-5

4.0 OVERVIEW
   4.1 System Reliability Prediction and Estimation Program. 4-1
      4.1.1 System Modeling. 4-1
      4.1.2 System Reliability Allocation. 4-7
      4.1.3 System Reliability Prediction. 4-8
      4.1.4 System Reliability Growth. 4-8
      4.1.5 System Reliability Qualification Testing. 4-9
      4.1.6 System-Level Failure Reporting and Corrective Action System (FRACAS). 4-9
   4.2 Hardware Reliability Prediction and Estimation Program 4-9
      4.2.1 Hardware Reliability Modeling 4-9
      4.2.2 Hardware Reliability Allocation. 4-10
      4.2.3 Hardware Reliability Prediction. 4-10
      4.2.4 Hardware Reliability Growth. 4-10
      4.2.5 Hardware Reliability Qualification Testing. 4-10
      4.2.6 Hardware FRACAS. 4-11
   4.3 Software Reliability Prediction And Estimation program 4-11
      4.3.1 Software Reliability Modeling 4-13
      4.3.2 Software Reliability Allocation 4-13
      4.3.3 Software Reliability Prediction 4-14
      4.3.4 Software Metrics Collection 4-16
      4.3.5 Software Reliability Growth Testing 4-18
      4.3.6 Software Reliability Qualification Testing 4-19
      4.3.7 Software Failure Reporting and Corrective Action System (FRACAS) 4-21
5.0 HARDWARE/SOFTWARE SYSTEM RELIABILITY MODELING 5-1

5.1 Basic Reliability Model 5-1

5.2 Mission Reliability Model 5-1

5.2.1 System FMEA Development 5-3

5.2.2 System-Level Reliability Model Development 5-5

5.2.3 Developing Detailed Reliability Models 5-5

5.2.4 Reliability Modeling of Hardware/Software Elements 5-6

5.2.4.1 Modeling Series Hardware/Software Elements

5-6

5.2.4.2 Modeling Redundant Hardware/Software Elements

5-7

5.2.4.2.1 Redundant Hardware Models 5-7

5.2.4.2.2 Redundant HW/SW Models 5-9

5.2.4.2.2.1 Cold Standby Systems 5-10

5.2.4.2.2.2 Hot Standby Systems 5-11

5.2.5 Hardware Failure and Repair Rates 5-13

5.2.6 Software Failure Rates 5-13

5.2.6.1 Timing Configurations 5-13

5.2.6.2 Reliability Topology 5-14

5.2.6.3 Notation 5-14

5.2.6.4 Software Failure Rate Adjustment 5-14

5.2.6.5 SW Reliability Combination Models. 5-15

Procedure 5.2.6.5-1 - Sequentially active software model. 5-16

Procedure 5.2.6.5-2 - Concurrently active software model. 5-17

Procedure 5.2.6.5-3 - Mission oriented software combination model 5-18

6.0 RELIABILITY ALLOCATION. 6-1

6.1 System Reliability Allocation. 6-1

6.2 Hardware Reliability Allocation. 6-1

6.3 Software Reliability Allocation 6-1

Procedure 6.3-1 - Equal apportionment applied to sequential software CSCIs. 6-10

Procedure 6.3-2 - Equal apportionment applied to concurrent software CSCIs. 6-10

Procedure 6.3-3 - Optimized allocation based on system-mode profile 6-11

Procedure 6.3-4 - Allocation based on operational criticality factors. 6-14

Procedure 6.3-5 - Allocation based on complexity factors 6-15
Procedure 6.3-6 - Allocation based on achievable failure rates
6-18
Procedure 6.3-7 - Re-allocation based on predicted failure rates.
6-22
6.4 Hardware/Software Allocations. 6-24
7.0 PREDICTION  7-1
  7.1 Hardware Reliability Prediction  7-1
  7.2 Software Reliability Prediction  7-1
        Integration Techniques” Method  7-4
        7.2.1.1 Proposal, Pre-Contract or Requirements
        Phase Prediction  7-6
        7.2.1.2 Design phase prediction.  7-7
        7.2.1.3 Coding, Unit Testing and Integration Phases
    Prediction.  7-8
    7.2.2 Raleigh Model  7-9
    7.2.3 Industry Data  7-10
    7.2.4 Musa Prediction Method.  7-11
    7.2.5 Historical Data Collection.  7-15
  7.3 Use of Predictions for Project Planning and Control  7-15
    7.3.1 The RL-TR-92-52 Model  7-15
    7.3.2 The Raleigh Model  7-15
    7.3.3 Industry Metrics Used  7-16
    7.3.4 The Musa Reliability Growth Method  7-18
  7.4 Forecasting Failure Rate Versus Execution Time 7-19
  7.5 Forecasting Cumulative Failures Versus Execution 7-19
  7.6 Forecasting When a Reliability Objective Will Be Met 7-19
  7.7 Additional Failure and Execution Time Required to Met and
    Objective 7-21
  7.8 Optimal Release Time  7-22
  7.9 Ultra High Reliability Prediction  7-24

8.0 RELIABILITY GROWTH AND DEMONSTRATION TESTING  8-1
  8.1 Software Operational Profile  8-1
  8.2 Random Input-State Selection  8-3
  8.3 Multiple Copies  8-4
  8.4 Software Reliability Growth Modeling/Testing  8-5
    8.4.1 A Checklist of Software Reliability Growth Models
    8-5
    8.4.2 - Goodness-of-fit/recalibration. 8-13
    8.4.3 Collecting the Data Required for the Models 8-13
  8.5 Software Reliability Demonstration  8-14
    Procedure 8.5-1 - Demonstration test 8-17
9.0 OPERATIONAL PROFILES  9-1
  9.1 Customer Profile  9-2
  9.2 User Profile ....  9-2
  9.3 System Mode Profile  9-2
  9.4 Functional Profile  9-3
    Procedure 9.4.1 - Generating a functional profile  9-4
  9.5 Operational Profile  9-7
  9.6 Operational Profile Development from Object-Oriented
Analysis/Design  9-16

APPENDIX
A.0 Impact of Design and Coding Techniques on Software Reliability  A-1
  A.1 The Link Between Software Reliability and Software Safety  A-7
    A.1.1 Use of Hypothesis Testing for Software Safety  A-7
    A.1.2. Ultra High Reliability and Safety  A-7
    A.1.3 Picking Test Cases for Safety-critical Real-time
Systems  A-9
    A.1.4 Safety Analyses A-9
    A.1.5 Software FMEAs and Fault Tree Analyses  A-10
  A.2 Description of SEI CMM Model A-12
  A.3 Matrix of Software Reliability Skill Sets  A-12
  A.4 Differences Between Software and Hardware Reliability  A-15
    A.4.1 Does Software Reliability Make Sense?  A-16
    A.4.2. What Should be Measured?  A-17
    A.4.3 Software Failure Rate Cannot be Predicted From Failure
Rates Per Individual Lines or Software Components.  A-17
    A.4.4. The Finite State Machine Model of Programs  A-17
  A.5 Software Testability  A-18
  A.6 Computing Complexity  A-19
  A.7 Software Metrics  A-21
  A.8 Additional Information on the Keene-Cole Model  A-23
  A.9 Additional Information on the Musa Model  A-24
  A.10 Bibliography  A-41
TABLES

TABLE 4-1. Reliability Prediction and Estimation Tasks. 4-2
TABLE 4-2. Software Reliability Allocation Procedures 4-14
TABLE 4-3. Software Reliability Prediction Factors. 4-17
TABLE 4-4. Software Reliability Qualification Test Types. 4-20
TABLE 4-5. List of Known Fault Types 4-22
TABLE 4-6. Orthogonal Defect Classification 4-24
TABLE 5-1. Software Failure Modes 5-5
TABLE 5-2. Series Sequential Example 5-16
TABLE 5-3. Operational Profile 5-19
TABLE 5-4. Mission Phases 5-21
TABLE 5-5. Operational Modes 5-22
TABLE 5-6. Effective Operating Time in Modes 5-22
TABLE 5-7. Software CSCIs 5-22
TABLE 5-8. Operational Mode Failure Rates 5-22
TABLE 6-1. Software Functions By System Mode- Example 6-4
TABLE 6-2. Sample System Mode Profile 6-7
TABLE 6-3. Operational Profile Allocation Factors 6-13
TABLE 6-4. Complexity Procedures 6-16
TABLE 6-5. CSCI Characteristics 6-20
TABLE 6-6. Growth Model Quantities 6-21
TABLE 7-1. Software Reliability Prediction Techniques 7-3
TABLE 7-2. Prediction Techniques by Phase 7-3
TABLE 7-3. Summary of the RL-TR-92-52 Model 7-4
TABLE 7-4. Conversion Ratio from Fault Density to Failure Rate 7-5
TABLE 7-5. Proposal/ Pre-Contract/Analysis Phase Factors 7-6
TABLE 7-6. Design Phase Factors 7-7
TABLE 7-7. Coding/Unit Testing/Integration Phase Factors 7-8
TABLE 7-8. Industry Data Prediction Technique 7-11
TABLE 7-9. Code Expansion Ratios 7-13
TABLE 7-10. Using Metrics For Planning and Control 7-16
TABLE 7-11. Suggested Defect Removal Efficiencies for SEI CMM Levels 7-18
TABLE 7-12. Methods for Predicting Optimal Release Time 7-22
TABLE 8-1. Software Reliability Models 8-6
TABLE 8-2. Failure-Free Execution Interval Test Plans 8-17
TABLE 9-1 Sample Customer Profile 9-2
TABLE 9-2 System Mode Profile 9-3
TABLE 9-3 (a) Sample Implicit Operational Profile 9-4
TABLE 9-3 (b) Sample Explicit Operational Profile 9-4
TABLE 9-4 Sample Final Function List 9-5
TABLE 9-5 Sample Functional Profile Segment 9-5
TABLE 9-6 Sample Environmental Profile 9-6
FIGURES

FIGURE 4-1. System Reliability Tasks 4-3
FIGURE 4-2. Reliability Model for HW/SW Element 4-4
FIGURE 4-3. Dependency of Software CSUs 4-5
FIGURE 4-4. Block Diagram for Automobile ABS 4-6
FIGURE 4-5. Block Diagram for Missile Guidance System 4-6
FIGURE 4-6. Block Diagram for Gateway Server System 4-7
FIGURE 4-7. Software Reliability Prediction Procedure 4-15
FIGURE 4-8. Software Failure Intensity Curve 4-18
FIGURE 4-9. Software FRACAS 4-21
FIGURE 5-1. Example of System-Level Functional FMEA 5-4
FIGURE 5-2. General Hardware Redundancy Model 5-7
FIGURE 5-3. Hardware Reliability Model 5-8
FIGURE 5-4. HW/SW Reliability Model 5-10
FIGURE 5-5. Simplified State Diagram 5-12
FIGURE 6-1. Event Diagram for Reliability Allocation 6-2
FIGURE 6-2. Basic Execution Time Software Reliability Model 6-5
FIGURE 6-3. Reliability Allocation Procedures 6-9
FIGURE 7-1. Raleigh Curve 7-10
FIGURE 8-1. Software Reliability Growth Models 8-9
FIGURE 8-2. Failure Rate Profiles 8-10
FIGURE 8-3. Failure Rate Curves 8-11
FIGURE 9-1 Operational Profile Development 9-1
FIGURE 9-2 Operational Elements 9-7
FIGURE 9-3 Plot of Selected Parameters from Free-flight Operational Profile 9-14
FIGURE 9-4 Vending Machine Object Representation 9-16
FIGURE 9-5 Stereo System Use Cases 9-16
FIGURE 9-6 Documented Use Case 9-17
FIGURE 9-7 Event Trace Example 9-17
1.0 INTRODUCTION

1.1 Purpose.

This notebook establishes uniform reliability assurance methods for predicting and estimating the reliability of electronic systems that include software components. It complements other reliability practices used in industry today.

1.2 Application.

This notebook provides both general requirements and specific procedures for predicting and estimating the reliability of systems that contain both hardware and software elements. Techniques are described for reliability modeling, allocation, prediction, growth modeling/testing, and qualification testing.

Section 4 provides a general overview of the system reliability tasks. Section 5 has techniques for modeling system software reliability. Section 6 provides guidance for allocating reliability to hardware and software components. Methods for predicting software and system reliability are discussed in Section 7. Section 8 discusses growth and demonstration testing. The software operational profile and how it impacts software reliability is presented in Section 9.0.

The appendix has additional information on safety analyses, measuring software complexity, organizational considerations with respect to software reliability, the Software Engineering Institute Capability Maturity Model, the key differences between software and hardware reliability, establishing a software metrics program, and reliability growth models.
2.0 APPLICABLE DOCUMENTS


Rentschler, D., Implementing Orthogonal Defect Classification, Transactions from the Fifth International Conference on Software Quality, October 1995, pages 277-279.


Voas, Jeffrey; Friedman, Michael, Software Assessment: Reliability, Safety and Testability, John Wiley & Sons, NY, 1995.
3.0 DEFINITIONS AND SYMBOLS

3.1 Definitions of Terms.

Aggregate. A generic term used to represent a collection of interrelated hardware and/or software components. An aggregate can exist at any level of the system structure. The hardware and/or software components that compose the aggregate exist at the next level below the aggregate.

Causal Analysis. Establishment of the root cause of a fault after it has been isolated and removed.

Component. A generic term used to represent a hardware or software item at any level in the system hierarchy.

Computer Software or Software Program. A combination of associated computer instructions and computer data definitions required to enable the computer hardware to perform computational or control functions.

Computer Software Component. A distinct part of a Computer Software Configuration Item (CSCI). CSCs may be further decomposed into other CSCs and Computer Software Units (CSUs).

Computer Software Configuration Item (CSCI). A configuration item that is computer software.

Configuration Item. An aggregation of hardware or software that satisfies an end use function and may be designated by the customer for separate configuration management.

Failure Rate. The rate at which failures occur in some interval. Failures per unit time.

Functional Baseline (FBL). The initially approved documentation describing a system’s functional, interoperability, and interface characteristics and the verification required to demonstrate the achievement of those specified characteristics.

Functional Profile. A software program's functional profile is a description of end-user functions and their probabilities of occurrence (proportion of time executed).

Hardware Configuration Item (HWCI). A configuration item that is hardware.

Hardware Failure. A hardware failure is the inability of a hardware item to perform a required function within specified limits.
Hazard Rate. The limit of the failure rate as the interval approaches zero; the instantaneous rate of failure at time $t$, given that the system survives until time $t$.

Inherent faults. The estimated total number of faults existing in the operational software, either observed or not.

Input Space. The input space is the set of all possible input states for a software program.

Input State. An input state is the set of values of input variables used by a software run.

Input Variable. An input variable is a data item that exists external to a run and is used by the run. There is one value for each variable for each run.

Mean Time to Software Restore (MTSWR). The amount of time needed to restore software operations on site. This is not the amount of time required to make a permanent repair to the software.

Non-Developmental Software (NDS). Deliverable software that is not developed under the contract but is provided by the contractor, the Government, or a third party. NDS may be referred to as reusable software, Government furnished software, or commercially available software, depending on the source.

Operating System. An operating system is the set of software products that jointly control the system resources and the processes using these resources. As used in this notebook, the term operating system includes both large, multi-user, multi-process operating systems and small real-time executives providing minimal services.

Output State. An output state is the set of values of output variables generated by a run.

Output Variable. An output variable is a data item that exists external to a run and is set by the run.

Per-Fault Hazard Rate. A per-fault hazard rate is the contribution each fault in a program makes to the overall program failure rate, when it is assumed that they contribute equally.

Product Baseline (PBL). The initially approved documentation describing all of the necessary functional and physical characteristics of the configuration item and the selected functional and physical characteristics designated for production acceptance.
testing and tests necessary for support of the configuration item. In addition to this
documentation, the product baseline of a configuration item may consist of the actual
equipment and software.

**Release.** The designation by the contractor that a document is complete and suitable for
use. Release means that the document is subject to the contractor’s configuration
control procedures.

**Re-Used Code.** Reused code is non-developmental software (NDS).

**Run.** A run is a result of the execution of a software program. A run has identifiable
input and output variables. The set of runs map the input space to the output space
and encompasses the software program’s operational profile.

**Software Defect.** A product anomaly that exists after the development activity in which
it was generated.

**Software Engineering Environment.** The set of automated tools, firmware devices, and
hardware necessary to perform the software engineering effort. The automated tools
may include but are not limited to compilers, assemblers, linkers, loaders, operating
system, debuggers, simulators, emulators, test tools, and database management
systems.

**Software Error.** A human action that results in software containing a fault.

**Software Fault.** A manifestation of an error in the software. If encountered, may cause
a failure.

**Software Failure.** An event in which a system or system component does not perform
a required function within specified limits.

**Software Metric.** A software metric is a measurable characteristic of the software
development process or of a work product of the development process.

**Software Operational Environment.** The manner in which the software will be operated
in its target environment.

**Software Operational Profile.** A quantitative characterization of how a system will be
used. A set of disjoint alternatives with the probability that each will occur.

**Software Reliability.** The probability that software will not cause the failure of a system
for a specified time under specified conditions. The probability is a function of the
inputs to, and use of, the system as well as a function of the existence of faults in the
software. The inputs to the system determine whether existing faults, if any, are encountered.

**System Reliability.** System reliability is the probability that a system, including all hardware and software subsystems, will perform a required task or mission for a specified time in a specified operational environment.

**Test Case.** A test case is a defined input state for a run, along with the expected output state.

**Version.** An identified and documented body of software.
3.2 Abbreviations.

cdf  cumulative distribution function
CPU  Central Processing Unit
CSC  Computer Software Component
CSCI  Computer Software Configuration Item
CSU  Computer Software Unit
ETT  Expected Test Time
FMEA  Failure Modes and Effects Analysis
FOM  Force of Mortality
FRACAS  Failure Reporting Analysis and Corrective Action System
FRB  Failure Review Board
HW  Hardware
HWCI  Hardware Configuration Item
KSLOC  (1000) Executable Source Lines of Code
LOC  Executable Source Lines of Code
MIPS  Million Instructions per Second
MTBF  Mean Time Between Failures
MTTF  Mean Time To Failure
NDS  Non-Developmental Software
NHPP  Non-Homogeneous Poisson Point
ODC  Orthogonal Defect Classification
OS  Operating System
pdf  probability density function
PRST  Probability Ratio Sequential Test
SDD  Software Design Document
SRS  Software Requirements Specification
SW  Software
TAAF  Test, Analyze, and Fix
3.3 Mathematical Symbols.

A application factor prediction parameter

B fault reduction factor

c_i criticality factor of the i\textsuperscript{th} CSCI

C computer time resource index

Cd fault detection coverage is the probability of detecting a fault given that a fault has occurred.

Ci fault Isolation coverage is the probability that a fault will be correctly isolated to the recoverable interface (level at which redundancy is available) given that a fault has occurred and been detected.

Cr fault recovery coverage is the probability that the redundant structure will recover system services given that a fault has occurred, been detected, and correctly isolated.

CP customer profile occurrence probability

D development factor prediction parameter

D() total cost

D_1() total system test failures cost

D_2() total operation failures cost

D_3() total system test cost

E_0 expected total failures in infinite time - equivalent to \( \nu_0 \) and \( N \)

E_c cumulative number of faults corrected

E_m estimated faults detected per month

E_T estimated total number of faults to be found during development and testing

E\{x\} expected value of x
exp[x]  exponential function: $e^x$

f  linear execution frequency

$f_i$  total expected failures detected in some interval $I$

F resource index of failure resolution personnel; total number of detected failures during test

FD fault density prediction

FP functional profile occurrence probability

F(x) cumulative distribution function

G*(x) recalibration function

I number of object instructions; resource index of failure identification personnel; input space

I_s number of source instructions.

I() Fisher information

K fault exposure ratio

ln x natural logarithm of $x$

M number of operational modes during a mission

$m_e$ cumulative number of failures during system test

N number of components in an aggregate; expected total failures in infinite time - equivalent to $\nu_0$

$p(i)$ probability of input state $i$

Q operational mode utilization matrix

$q_{ij}$ fraction of time that $j$th mode is utilized during $i$th phase

$R_H$ reliability of the hardware
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rs</td>
<td>reliability of the software</td>
</tr>
<tr>
<td>Rsys</td>
<td>reliability of a system</td>
</tr>
<tr>
<td>R(t)</td>
<td>reliability function with respect to time</td>
</tr>
<tr>
<td>SA</td>
<td>software anomaly management prediction parameter</td>
</tr>
<tr>
<td>SL</td>
<td>software language prediction parameter</td>
</tr>
<tr>
<td>SLOCj</td>
<td>number of SLOC in a component j</td>
</tr>
<tr>
<td>SM</td>
<td>system mode occurrence probability; software modularity prediction parameter</td>
</tr>
<tr>
<td>SQ</td>
<td>software quality review prediction parameter</td>
</tr>
<tr>
<td>SR</td>
<td>software standard review prediction parameter</td>
</tr>
<tr>
<td>ST</td>
<td>software traceability prediction parameter</td>
</tr>
<tr>
<td>SX</td>
<td>software complexity prediction parameter</td>
</tr>
<tr>
<td>r</td>
<td>resource index (C, I, or F); average instruction execution rate</td>
</tr>
<tr>
<td>T</td>
<td>mission phase duration matrix</td>
</tr>
<tr>
<td>t</td>
<td>generic time; calendar time since the beginning of system test</td>
</tr>
<tr>
<td>tD</td>
<td>total development and test calendar time</td>
</tr>
<tr>
<td>U</td>
<td>utilization matrix</td>
</tr>
<tr>
<td>US</td>
<td>user probability occurrence probability</td>
</tr>
<tr>
<td>V</td>
<td>number of phases in a mission</td>
</tr>
<tr>
<td>wi</td>
<td>complexity weighting factor of ith CSCI</td>
</tr>
<tr>
<td>X</td>
<td>effective operating time matrix</td>
</tr>
<tr>
<td>z(t)</td>
<td>instantaneous failure rate at time t</td>
</tr>
</tbody>
</table>
$\alpha$ confidence level; producer’s risk
$\beta$ decrement of failure rate per failure experienced; consumer’s risk
$\delta$ discrimination ration used for reliability demonstration testing
$\kappa$ normal deviate
$\Lambda$ average aggregate failure rate over an interval
$\Lambda_G$ failure rate goal of an aggregate
$\Lambda_P$ predicted average aggregate failure rate over an interval
$\lambda$ constant failure rate
$\lambda_0$ initial failure rate (the software failure rate at the start of system test)
$\lambda_F$ future failure rate objective
$\lambda_i$ failure rate of the i-th component in an aggregate
$\lambda_{iG}$ failure rate goal of the i-th component in an aggregate
$\lambda_P$ present failure rate
$\lambda_{iP}$ predicted failure rate of the i-th component in an aggregate.
$\lambda(t)$ failure rate at time t.
$\mu$ number of copies of software concurrently operating
$\mu_r$ failure coefficient of resource r usage
$\mu(t)$ mean value function: expected number of failures experienced by time t
$\nu_0$ expected total failures in infinite time
$\theta$ MTBF, average failure effort per resource
$\rho$ average number of occurrences of a single fault; fault density prediction
\( \rho \) utilization of personnel resource

\( \tau \) cumulative execution time since the beginning of system test

\( \tau_e \) cumulative execution time into system test, at which software is actually or hypothetically released.

\( \tau_i \) cumulative execution time at which i-th failure occurs

\( \tau' \) execution time measured from present

\( \tau'_i \) i-th interfailure time; active time of i-th component

\( \phi \) per-fault hazard rate

\( \omega_0 \) number of inherent faults (the number of faults in the code at the start of system test)

\( \omega_{0i} \) number of inherent faults in i-th CSCI

\( \xi \) failure rate adjustment

\( \Psi \) reliability growth estimate