SYSTEM AND SOFTWARE RELIABILITY ASSURANCE NOTEBOOK

Produced For Rome Laboratory
By

Peter B. Lakey, McDonnell Douglas Corporation, St. Louis, MO
Ann Marie Neufelder, SoftRel, Hebron, KY

FSC-RELI

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.

System and Software Reliability Assurance

Table of Contents

APPENDIX

TABLES

FIGURES

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

American Institute of Aeronautics and Astronautics, Recommended Practice for Software Reliability ANSI/AIAA R-013-1992, February 23, 1993.

Chillarege, Ram, Orthogonal Defect Classification - A Concept for in-Process Measurements, IEEE Transactions on Software Engineering, 11/92.

Farr, Dr. William, A Survey of Software Reliability Modeling and Estimation, NSWC TR 82-171, Naval Surface Weapons Center, Dahlgren, VA, Sept. 1983.

Friedman, M.A., Tran, P.Y., and Goddard, P.L., Reliability Techniques for Combined Hardware and Software Systems, Final Report, Contract F30602-89-C-0111, Rome Laboratory, Air Force Systems Command, Griffiss Air Force Base, New York. Sept. 1991.

Jones,Capers, "Backfiring" or Converting Lines of Code Metrics Into Function Points, Software Productivity Research, Burlington, MA, October 6, 1995.

Jones, Capers, Measuring Global Software Quality, Software Productivity Research, Burlington, MA, 1995.

Jones, Capers, Software Productivity Research, Inc., Applied Software Measurement, McGraw-Hill, NY, 1995.

Keene, Dr. Samuel, Cole, G.F., Reliability Growth of Fielded Software, Reliability Review, Vol 14, March 1994.

Lyu, Michael R., Handbook of Software Reliability Engineering, IEEE Computer Society Press, 1996.

Musa, J.D., Iannino, A. and Okumoto, K., Software Reliability: Measurement, Prediction, Application, McGraw Hill Book Company, New York, NY. 1987.

Musa, J.D., Operational Profiles in Software Reliability Engineering, IEEE Software Magazine, March 1993, pages 14-32.

Parnas, David L., Evaluation of Safety-Critical Software, Communications of the ACM, Vol. 33, No. 6, June 1990.

Putnam, L., Myers W., Measures for Excellence, Prentice Hall Yourdon Press, Englewood Cliffs, NJ, 1992.

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

Rook, P., Software Reliability Handbook, Centre for Software Reliability, Elsevier Applied Science, London, 1990.

Science Applications International Corporation & Research Triangle Institute, Software Reliability Measurement and Testing Guidebook, Final Technical Report, Contract F30602-86-C-0269, Rome Air Development Center, Griffiss Air Force Base, New York, January 1992.

SEMATECH, Tactical Software Reliability Guidebook, Technology Transfer #95092967A-GEN, Fulton, S.; Neufelder, AM, , Austin, Tx, 1995.

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^{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 ν_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 ν_0
p(i)	probability of input state i
Q	operational mode utilization matrix
q_{ij}	fraction of time that jth mode is utilized during ith phase
R_H	reliability of the hardware
R_S	reliability of the software

R_{SYS}	reliability of a system
R(t)	reliability function with respect to time
SA	software anomaly management prediction parameter
SL	software language prediction parameter
$SLOC_j$	number of SLOC in a component j
SM	system mode occurrence probability; software modularity prediction parameter
SQ	software quality review prediction parameter
SR	software standard review prediction parameter
ST	software traceability prediction parameter
SX	software complexity prediction parameter
r	resource index (C, I, or F); average instruction execution rate
T	mission phase duration matrix
t	generic time; calendar time since the beginning of system test
t_D	total development and test calendar time
U	utilization matrix
US	user probability occurrence probability
V	number of phases in a mission
w_i	complexity weighting factor of i^{th} CSCI
X	effective operating time matrix
z(t)	instantaneous failure rate at time t
α	confidence level; producer's risk

β	decrement of failure rate per failure experienced; consumer's risk
δ	discrimination ration used for reliability demonstration testing
κ	normal deviate
Λ	average aggregate failure rate over an interval
Λ_G	failure rate goal of an aggregate
Λ_P	predicted average aggregate failure rate over an interval
λ	constant failure rate
λ_0	initial failure rate (the software failure rate at the start of system test)
λ_F	future failure rate objective
λ_i	failure rate of the i-th component in an aggregate
λ_{iG}	failure rate goal of the i-th component in an aggregate
λ_P	present failure rate
λ_{iP}	predicted failure rate of the i-th component in an aggregate.
$\lambda(t)$	failure rate at time t.
μ	number of copies of software concurrently operating
μ_r	failure coefficient of resource r usage
$\mu(t)$	mean value function: expected number of failures experienced by time t
ν_0	expected total failures in infinite time
θ	MTBF, average failure effort per resource
ρ	average number of occurrences of a single fault; fault density prediction
ρ_r	utilization of personnel resource

τ	cumulative execution time since the beginning of system test
τ_e	cumulative execution time into system test, at which software is actually or hypothetically released.
τ_i	cumulative execution time at which i-th failure occurs
τ'	execution time measured from present
τ'_i	i-th interfailure time; active time of i-th component
ϕ	per-fault hazard rate
ω_0	number of inherent faults (the number of faults in the code at the start of system test)
ω_{0i}	number of inherent faults in i-th CSCI
ξ	failure rate adjustment
Ψ	reliability growth estimate

â€" Attributes often conflict â€¢ Systems and software reliability in a competitive world: values and tradespaces â€¢ Conclusions Fall 2015 Â©USC-CSSE 2 USC C S E University of Southern California Center for Software Engineering Software Reliability Business Case â€¢ Systems and software reliability investments compete for resources â€" With investments in functionality, response time, adaptability, speed of developmentÂ Protection: safety, security, privacy Robustness: reliability, availability, survivability Quality of Service: performance, accuracy, ease of use Adaptability: evolvability, interoperability Affordability: cost, schedule, reusability, maintainability â€¢ Value attributes Software Quality Assurance in Large Scale and Complex Software-intensive Systems presents novel and high-quality research related approaches that relate the quality of software architecture to system requirements, system architecture and - Selection from Software Quality Assurance [Book].Â Software Quality Assurance. by Bedir Tekinerdogan, John Grundy, Nour Ali, Richard M Soley, Ivan Mistrik. Publisher: Morgan Kaufmann. Definitions of Software Reliability Factors Influencing Software Reliability Applications of Software Reliability Operational Profiles Reliability Models Summary. Slide 3. 3 What is Reliability?Â 17 Reliability Models Example Assume that a software system is undergoing system level testing. The initial failure intensity of the system was 25 failures/CPU hours, and the current failure intensity is 5 failures/CPU hour. It has been decided by the project manager that the system will be released only after the system reaches a reliability level of at most 0.001 failures/CPU hour.