

IEEE Standard Dictionary of Measures to Produce Reliable Software

IEEE Standards Board

Approved June 9, 1988

American National Standards Institute

Approved August 10, 1989

Sponsor

**Software Engineering Standards Subcommittee of the Technical Committee on
Software Engineering
of the
IEEE Computer Society**

© Copyright 1989 by **The Institute of Electrical and Electronics Engineers, Inc 345 East 47th Street,
New York, NY 10017-2394, USA**

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

PDF: ISBN 0-7381-0397-7, SS12542

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least once every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board 345 East 47th Street New York, NY 10017 USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Foreword

(This Foreword is not a part of IEEE Std 982.1-1988, IEEE Standard Dictionary of Measures to Produce Reliable Software.)

This standard provides a set of measures indicative of software reliability that can be applied to the software product as well as to the development and support processes. It was motivated by the need of software developers and users who are confronted with a plethora of models, techniques, and measures. There is a need for measures that can be applied early in the development process that may be indicators of the reliability of the delivered product. This standard provides a common, consistent definition of a set of measures that may meet those needs.

This standard is intended to serve as the foundation on which researchers and practitioners can build consistent methods. This document is designed to assist management in directing product development and support toward specific reliability goals. Its purpose is to provide a common set of definitions through which a meaningful exchange of data and evaluations can occur. Successful application of the measures is dependent upon their use as intended in the specific environments.

There is both a process goal and a product goal for this document:

- 1) The process goal is to provide measures that may be applicable throughout the life cycle and may provide the means for continual self-assessment and reliability improvement.
- 2) The product goal is to increase the reliability of the software in its actual use environment during the operations and support phases.

This standard is intended to be of interest to design, development, evaluation (for example, auditors, procuring agency) and maintenance personnel; software quality and software reliability personnel; and to operations and acquisition support managers. This document contains four sections:

Section 1 : Scope, establishes the goals and boundaries of the standard.

Section 2 : Definitions, serves as a central location for key terms used throughout the body of the document.

Section 3 : Functional Classification of Measures, provides a taxonomy with respect to measure objectives.

Section 4 : Measures for Reliable Software, presents the measures ordered in general by complexity.

Introduction

The Software Reliability Measurement Working Group Steering Committee had the following membership:

	James Dobbins, <i>Chair</i>	
	Ray Leber, <i>Chair</i>	
	Ted Workman, <i>Chair</i>	
Peter Bright	Laura Good	Frank Salvia
Antonio Cicu	Nancy Hall	David Siefert
Carlo Cucciati	Philip Marriott	Raghu Singh
Walter Ellis	Roger Martin	Vijaya Srivastava
William Farr	Walter Murch	Henry Trocheset
Stuart Glickman	Patricia Powell	William Wilson

The Software Reliability Measurement Working Group had the following membership:

B. Andrews	W. Goss	K. Oar
S. Beason	S. Gray	R. Panchal
R. Berlack	F. Gray	R. Pikul
M. Berry	V. Haas	E. Presson
N. Beser	C. Harrison	R. Prudhomme
J. Bieman	S. Hartman	H. Richter
J. Blackman	L. Hearn	M. Richter
J. Bowen	H. Hecht	L. Sanders
M. Bruyere	R. Hood	R. Sandborgh
F. Buckley	J. Horch	H. Schock
N. Chu	D. Hurst	D. Simkins
G. Chisholm	N. Johnson, Jr	O. Smith
G. Clow	R. Kenett	R. Spear
J. Corrinne	R. Kessler	E. Soistman
W. Covington	R. Kumar	L. Sprague
W. Daney	M. Landes	A. Stone
B. Daniels	R. Lane	D. Sudbeck
F. DeKalb	J. Latimer	R. Sulgrove
I. Doshay	I. Lebowitz	D. Town
E. Dunaye	M. Lipow	R. Wamser
F. Fisk	C. McPherson	J. Wang
A. Florence	A. Moseley	D. Winterkorn
D. Frevert	S. Nemecek	J. Wujek
D. Gacke	K. Nidiffer	B. Zamastil

The following persons were the balloting committee that approved this document for submission to the IEEE Standards Board:

F. Ackerman	J. Forster	R. Martin
W. Boll	D. Gelperin	J. McCall
J. Bowen	S. Glickman	J. Mersky
P. Bright	S. Gloss-Sloer	W. Murch
F. Buckley	L. Good	J. Navlakha
H. Carney	D. Gustavson	D. Nickle
J. Chung	V. Haas	D. Peercy
F. Coallier	C. Hay	W. Perry
P. Denny	H. Hecht	P. Petersen
J. Dobbins	L. Kaleda	D. Pfeiffer
D. Doty	T. Kurihara	R. Poston
R. Dwyer	G. Larsen	P. Powell
W. Ellis	F. C. Lim	N. Schneidewind
W. Farr	B. Lindberg	W. Schnoege
D. Favor	M. Lipow	R. Schueppert
J. Fendrich	B. Livson	R. Shillato
G. Fields	P. Marriott	D. Siefert

D. Simkins
R. Singh
V. Srivastava
A. Sukert
K. Tai

R. Thibodeau
P. Thompson
D. Town
H. Trochesset
R. van Tilburg

P. Wolfgang
T. Workman
N. Yopconka

When the IEEE Standards Board approved this standard on June 9, 1988, it had the following membership:

Donald C. Fleckenstein, *Chair*
Macro Migliaro, *Vice Chair*
Andrew G. Salem, *Secretary*

Arthur A. Blaisdell
Fletcher J. Buckley
James M. Daly
Stephen R. Dillon
Eugene P. Fogarty
Jay Forster*
Thomas L. Hannan
Kenneth D. Hendrix
Theodore W. Hissey, Jr.

John W. Horch
Jack M. Kinn
Frank D. Kirschner
Frank C. Kitzantides
Joseph L. Koepfinger*
Irving Kolodny
Edward Lohse
John E. May, Jr.
Lawrence V. McCall

L. Bruce McClung
Donald T. Michael*
Richard E. Mosher
L. John Rankine
Gary S. Robinson
Frank L. Rose
Helen M. Wood
Karl H. Zaininger
Donald W. Zipse

*Member Emeritus

The following organizations supported the development of this standard:

Aerospace Corporation
Apple Computer, Inc.
Argonne National Laboratory
Bechtel Group, Inc.
Bell Canada
Boeing Aerospace
Computer Sciences Corp.
CTA, Inc.
Digital Equipment Corp.
Dynamics Research Corp.
EG&G
Federal Aviation Administration
General Dynamics
General Electric
Hewlett-Packard
Honeywell
Hughes Aircraft Co.
IBM
INTEL Corporation
International Bureau Software Test
ITT Telecom
Martin-Marietta
McDonnell Douglas
NASA
National Bureau of Standards
National Centre of Systems Reliability
NCR Corporation
Norden Systems
PRC
Product Assurances Consulting
RCA

Sanders Associates
Satellite Business Systems
SOFTECH Microsystems
SoHaR, Incorporated
Sperry
Stromberg Carlson
Tadiran, Ltd.
Tandem Computers
Teledyne Brown Engineering
Transport Canada
TRW
U.S. Air Force
U.S. Army
U.S. Navy
University of Southwestern Louisiana
Xerox Corporation

Contents

CLAUSE	PAGE
1. Introduction.....	1
1.1 Scope.....	1
1.2 References.....	1
2. Definitions.....	2
3. Functional Classification of Measures.....	2
3.1 Product Measures.....	3
3.2 Process Measures	3
4. Measures for Reliable Software.....	5
4.1 Fault Density	5
4.2 Defect Density	5
4.3 Cumulative Failure Profile.....	6
4.4 Fault-Days Number	6
4.5 Functional or Modular Test Coverage	7
4.6 Cause and Effect Graphing	8
4.7 Requirements Traceability	9
4.8 Defect Indices	10
4.9 Error Distribution(s)	11
4.10 Software Maturity Index	13
4.11 Manhours per Major Defect Detected	14
4.12 Number of Conflicting Requirements	15
4.13 Number of Entries and Exits per Module	15
4.14 Software Science Measures	15
4.14 Software Science Measures	15
4.15 Graph-Theoretic Complexity for Architecture	16
4.16 Cyclomatic Complexity	18
4.17 Minimal Unit Test Case Determination	18
4.18 Run Reliability	19
4.18 Run Reliability	19
4.19 Design Structure	20
4.20 Mean Time to Discover the Next K Faults	21
4.21 Software Purity Level	21
4.22 Estimated Number of Faults Remaining (by Seeding)	22
4.23 Requirements Compliance	23
4.24 Test Coverage	24
4.25 Data or Information Flow Complexity	25
4.26 Reliability Growth Function	26
4.27 Residual Fault Count	27
4.28 Failure Analysis Using Elapsed Time	27
4.29 Testing Sufficiency	28
4.30 Mean-Time-to-Failure	29
4.31 Failure Rate	29
4.32 Software Documentation and Source Listings	30

4.33 RELY (Required Software Reliability)	31
4.34 Software Release Readiness	33
4.35 Completeness	35
4.36 Test Accuracy	36
4.37 System Performance Reliability	37
4.38 Independent Process Reliability	38
4.39 Combined Hardware and Software (System) Operational Availability	39

List of Symbols

Symbol	Meaning	Measures Used	Primitive/Metric
A	(1) Number of arrivals during time period T (2) Reliability growth factor	37 26	P M
A_{existing}	Number of ambiguities in a program remaining to be eliminated	6	P
$A(t)$	System availability	39	M
A_{tot}	Total number of ambiguities identified	6	P
α	Test accuracy	36	M
AQ	Average que length	37	M
AS	Average service time	37	P
B	Software science measure (number of errors)	14	M
B_1	Number of functions not satisfactorily defined	35	P
B_2	Number of functions	35	P
B_3	Number of data references not having an origin	35	P
B_4	Number of data references	35	P
B_5	Number of defined functions not used	35	P
B_6	Number of defined functions	35	P
B_7	Number of referenced functions not defined	35	P
B_8	Number of referenced functions	35	P
B_9	Number of decisions points not using all conditions, options	35	P
B_{10}	Number of decision points	35	P
B_{11}	Number of condition options without processing	35	P
B_{12}	Number of condition options	35	P
B_{13}	Number of calling routines with parameters not agreeing with defined parameters	35	P
B_{14}	Number of calling routines	35	P
B_{15}	Number of condition options not set	35	P
B_{16}	Number of set condition options having no processing	35	P
B_{17}	Number of set condition options	35	P
B_{18}	Number of data references having no destination	35	P
β	Observed hardware failure rate	39	P

Symbol	Meaning	Measures Used	Primitive/Metric
C	Complexity metric (static, dynamic, cyclomatic)	15, 16	M
c_i	Complexity for program invocation and return along each edge e_i as determined by user	15	P
CE	Cause and effect measure	6	M
CM	Completeness measure	6	M
D	Software science measure (program difficulty)	14	M
d_k	Complexity for allocation of resource k as determined by the user	15	P
D_i	Total number of unique defects detected during the i th design or code inspection process or the i th life cycle process	2, 8	P
DD	Defect density	2	M
DEs	Decomposition elements	23	M
DI	Defect index	8	M
DSM	Design structure metric	19	M
datain	Number of data structures from which the procedure retrieves data	25	P
dataout	Number of data structures that the procedure updates	25	P
E	(1) Number of edges (2) Software science measure (effort)	15, 16, 17 14	M
e_i	Number of entry points for the i th module	13	P
F or F_i	Total number of unique faults found in a given time interval resulting in failures of a specified severity level	1	P
f or f_i	Total number of failures of a given severity level in a given time interval	3, 20, 21, 27, 31	P
F_d	Fault density	1	M
f_{ij}	Frequency execution of intermodule transfer from module i to j	38	P
F_a	Number of software functions (modules) in the current delivery that are additions in the current delivery	10	P
F_c	Number of software functions (modules) in the current delivery that include internal changes from a previous delivery	10	P
FD	Fault days metric	4	M
FD_i	Fault days for the i th fault	4	P
F_{del}	Number of software functions (modules) in the previous delivery that are deleted in the current delivery	10	P

Symbol	Meaning	Measures Used	Primitive/Metric
FE	Number of the software functional (modular) requirements for which all test cases have been satisfactorily completed	5	P
F_{it}	Total number of faults detected to date in software integration testing	29	P
F_{pit}	Number of faults detected to date in software integration testing	29	P
FT	Total number of software functional (modular) test requirements	5	P
Γ	Average rate of jobs	37	M
Υ	Observed hardware repair rate	39	P
I	Total number of inspections to date	2, 11	P
IFC	Information flow complexity	25	M
J	Total number of jobs completed	37	P
K	Number of resources	15	P
k	Number of runs for the specified period of time	18	P
$KSLOC$	Number of source lines of executable code and nonexecutable data declarations in thousands	1, 2, 8	P
$KSLOD$	Number of source lines of design statements in thousands that have been inspected to date	2, 8	P
L	Software science measure (observed program length)	14	M
l	Software science measure (program vocabulary)	14	M
Ll	Software science measure (program level)	14	M
$\lambda(t)$	Failure rate function	31	M
λ	Observed software failure rate	39	P
length	Number of source statements in a procedure	25	P
lfi	Local flows into a procedure	25	P
lfo	Local flows from a procedure	25	P
M	Manhours per major defect detected	11	M
M_i	Number of medium defects found	8	P
m_i	Number of entry and exit points for module i	13	M
MD	Number of modules	38	P
$ME(i)$	Measure effectiveness for the i th issue	34	P
M_{it}	Number of modules integrated	29	M
M_T	Number of software functions (modules) in the current delivery	10	M

Symbol	Meaning	Measures Used	Primitive/Metric
M_{tot}	Total number of modules in final configuration	29	M
MTTF	Mean-time-to-failure	30	M
μ_i	Observed software fault correction rate with i faults in the software	39	P
N	Number of nodes	15, 16, 17	P
nc	Number of correct runs in a given test sample	18, 26	P
NF	Total number of faults within a program	29, 39	M
NF_{rem}	Total number of faults remaining within a program	22, 29	M
$NM_{rem}(\%)$	Percentage of faults remaining within a program	36	M
n_F	Number of faults found that were not intentionally seeded	22	P
NR	Number of runs (or test cases) in a given test sample	18, 26	P
NS	Total number of stages	26	P
N_s	Number of seeded faults	22, 36	P
n_s	Number of seeded faults found	22	P
$N1$	Total number of occurrences of the operators in a program	14	P
$n1$	Number of distinct operators in a program	14	P
$N2$	Total number of occurrences of the operands in a program	14	P
$n2$	Number of distinct operands in a program	14	P
N_1	Number of errors detected using SVDs due to inconsistencies	23	P
N_2	Number of errors detected using SVDs due to incompleteness	23	P
N_3	Number of errors detected using SVDs due to misinterpretation	23	P
P	Probability measure over the sample space	18	P
p_n	Probability of repairing a hardware failure correctly	39	P
P_i	(1) Probability that the i th run is selected from the sample space (2) Probability that the i th process that can be generated in a user environment is selected	18, 38 39	P P
P_s	Probability of correcting a software fault when detected	39	P
$P_{N,n}(t)$	System update probabilities	39	P
PI_i	Phase index metric for the i th life cycle phase	8	M

Symbol	Meaning	Measures Used	Primitive/Metric
PL	Purity level	21	M
PS	Product size	8	P
$P1$	Total number of modules in the program	19	P
$P2$	Number of modules dependent on the input or output	19	P
$P3$	Number of modules dependent on prior processing	19	P
$P4$	Number of database elements	19	P
$P5$	Number of nonunique database elements	19	P
$P6$	Number of database elements	19	P
$P7$	Number of modules not single entrance/single exit	19	P
Q	Que length distribution	37	P
q_i	Probability that P_i will be generated in a user environment	38	P
R	(1) Total response time for each functional job (2) Reliability of the system	37 38	M M
R_i	Reliability of i th module	38	P
R_k	Run reliability at a specified stage	18, 26	M
$R(t)$	Reliability function	31	M
r_i	Flag to denote if the i th process generates the correct software system output	38	P
r_k	Resource status array indicator flags	15	P
RG	Number of regions	16, 17	P
$RK(i)$	Risk factor for the i th measure	34	M
RT	Response time distribution	37	P
$R1$	Number of requirements met by the architecture	7	P
$R2$	Number of original requirements	7	P
S	Sample space of possible input patterns and states	18	P
S_i	Point from the sample space of possible inputs	18	P
s_i	Number of serious defects found	8, 11	P
SB	Total amount of time server is busy	37	P
SE	Server's efficiency	37	P
SN	Number of splitting nodes	16, 17	P
SMI	Software maturity index	10	M
ST	Service time distribution	37	P
T	(1) Time period over which measurements are made (2) Time	37 14	M

Symbol	Meaning	Measures Used	Primitive/Metric
T_i	Number of trivial defects found	8	P
t_i	Observed times between failures of a given severity level	20, 21, 27 28, 30, 31	P
TC	Test coverage	24	M
TM	Traceability measure	7	M
T_1	Time expended by the inspection team in preparation for design or code inspection meeting	11	P
T_2	Time expended by the inspection team in conduct of a design or code inspection meeting	11	P
U	Utilization	37	P
V	Software science measure (program volume)	14	M
VR	Number of requests per functional job for each server during time period T	37	P
W_i	Weighting distribution	8, 19, 34	P
WT	Waiting time distribution	37	P
X	Throughput	37	P
x_i	Number of exit points for the i th module	13	P

NOTE—P stands for primitive; M stands for derived metric

IEEE Standard Dictionary of Measures to Produce Reliable Software

1. Introduction

1.1 Scope

The objective of this standard is to provide the software community with defined measures currently used as indicators of reliability. The predictive nature of these measures is dependent on correct application of a valid model.

This standard presents a selection of applicable measures, the proper conditions for using each measure, the methods of computation, and the framework for a common language among users. It moreover provides the means for continual assessment of the process, as well as the products, at each phase of the life cycle. By emphasizing early reliability assessment, this standard supports methods through measurement to improve the product reliability. Successful application of the measures is dependent upon their use as intended in the specified environments.

Applicability is not restricted by the size, type, complexity, or criticality of the software. Typical life cycle phases based upon ANSI/IEEE Std 729-1983 1, ¹ are used to illustrate when each of the measures may be applied.

Errors, faults, and failures serve as primitive units for the majority of the measures.

1.2 References

The following publications shall be used in conjunction with this standard:

[1] ANSI/IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology. ²

[2] Software Maintainability - Evaluation Guide, AFOTEC Pamphlet 800-2, vol 3, Mar 1987. ³

¹ The numbers in brackets correspond to those of the references listed in 1.2 .

² ANSI/IEEE publications are available from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018, or from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08854-1331.

³ AFOTEC publications are available from the Air Force Operational Test and Evaluation Center (AFOTEC/DAP), Kirtland Air Force Base, NM 87117-7001.

2. Definitions

Most of the definitions listed below can be found in ANSI/IEEE Std 729-1983 1, IEEE Standard Glossary of Software Engineering Terminology. The definitions of the terms *concept phase*, *defect*, *measure*, *primitive*, and *software management* are not present in ANSI/IEEE Std 729-1983 1, and establish meaning in the context of this standard. The acronym ISO (International Organization for Standardization) indicates that the definition is also accepted by ISO.

concept phase: The period of time in the software life cycle during which system concepts and objectives needed by the user are identified and documented. Precedes the requirements phase.

defect: A product anomaly. Examples include such things as (1) omissions and imperfections found during early life cycle phases and (2) faults contained in software sufficiently mature for test or operation. See also **fault**.

error: Human action that results in software containing a fault. Examples include omission or misinterpretation of user requirements in a software specification, incorrect translation, or omission of a requirement in the design specification (ANSI/IEEE Std 729-1983 1).

failure: (1) The termination of the ability of a functional unit to perform its required function (ISO; ANSI/IEEE Std 729-1983 1). (2) An event in which a system or system component does not perform a required function within specified limits. A failure may be produced when a fault is encountered (ANSI/IEEE Std 729-1983 1).

fault: (1) An accidental condition that causes a functional unit to fail to perform its required function (ISO; ANSI/IEEE Std 729-1983 1). (2) A manifestation of an error in software. A fault, if encountered, may cause a failure. Synonymous with **bug** (ANSI/IEEE Std 729-1983 1).

measure: A quantitative assessment of the degree to which a software product or process possesses a given attribute.

primitive: Data relating to the development or use of software that is used in developing measures or quantitative descriptions of software. Primitives are directly measurable or countable, or may be given a constant value or condition for a specific measure. Examples include: error, failure, fault, time, time interval, date, number of noncommentary source code statements, edges, and nodes.

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 (ANSI/IEEE Std 729-1983 1).

software reliability management: The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources (cost), schedule, and performance.

3. Functional Classification of Measures

The measures can be divided into two functional categories: product and process (Table 3-1). Product measures are applied to the software objects produced and are divided into six subcategories. Process measures are applied to the activities of development, test, and maintenance. Process measures are divided into three subcategories.

3.1 Product Measures

The product measures address cause and effect of the static and dynamic aspects of both projected reliability prior to operation, and operational reliability. As an example, reliability may change radically during the maintenance effort, due to the complexity of the system design. These product measures cover more than the correctness aspect of reliability; they also address the system utility aspect of reliability. The following six product measure subcategories address these dimensions of reliability:

- 1) Errors; Faults; Failures—Count of defects with respect to human cause, program bugs, observed system malfunctions.
- 2) Mean-Time-to-Failure; Failure Rate—Derivative measures of defect occurrence and time.
- 3) Reliability Growth and Projection—The assessment of change in failure-freeness of the product under testing and in operation.
- 4) Remaining Product Faults—The assessment of fault-freeness of the product in development, test, or maintenance.
- 5) Completeness and Consistency—The assessment of the presence and agreement of all necessary software system parts.
- 6) Complexity—The assessment of complicating factors in a system.

3.2 Process Measures

The three process measure subcategories are directly related to process management:

- 1) Management Control—The assessment of guidance of the development and maintenance processes.
- 2) Coverage—The assessment of the presence of all necessary activities to develop or maintain the software product.
- 3) Risk; Benefit; Cost Evaluation—The assessment of the process tradeoffs of cost, schedule, and performance.

The measure classification matrix (Table 3-1) is a cross index of the measures and functional categories and can be used to select measures applicable to each category.