

This is a preview - [click here to buy the full publication](#)

INTERNATIONAL STANDARD

ISO/IEC 5055

First edition
2021-03

Information technology — Software measurement — Software quality measurement — Automated source code quality measures



Reference number
ISO/IEC 5055:2021(E)

© ISO/IEC 2021



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier; Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

1	Scope	1
1.1	Purpose	1
1.2	Overview of Structural Quality Measurement in Software.....	1
2	Conformance	2
3	Normative References.....	3
4	Terms and Definitions	4
5	Symbols (and Abbreviated Terms)	7
6	Weaknesses Included in Quality Measures and Representation Metamodels.....	8
6.1	Purpose	8
6.2	Software Product Inputs	8
6.3	Automated Source Code Quality Measure Elements.....	8
6.4	Automated Source Code Maintainability Measure Element Descriptions	9
6.5	Automated Source Code Performance Efficiency Measure Element Descriptions	11
6.6	Automated Source Code Reliability Measure Element Descriptions	15
6.7	Automated Source Code Security Measure Element Descriptions	23
6.8	Introduction to the Specification of Quality Measure Elements	32
6.9	Knowledge Discovery Metamodel (KDM).....	32
6.10	Software Patterns Metamodel Standard (SPMS).....	36
6.11	Reading guide.....	37
7	List of ASCQM Weaknesses.....	38
7.1	Weakness Category Maintainability	38
7.1.1	CWE-407 Algorithmic Complexity	38
7.1.2	CWE-478 Missing Default Case in Switch Statement.....	38
7.1.3	Weakness CWE-480 Use of Incorrect Operator	38
7.1.4	CWE-484 Omitted Break Statement in Switch	39
7.1.5	CWE-561 Dead Code	39
7.1.6	CWE-570 Expression is Always False	39
7.1.7	CWE-571 Expression is Always True	39
7.1.8	CWE-783 Operator Precedence Logic Error	40
7.1.9	CWE-1075 Unconditional Control Flow Transfer Outside of Switch Block	40
7.1.10	CWE-1121 Excessive McCabe Cyclomatic Complexity Value.....	40
7.1.11	CWE-1054 Invocation of a Control Element at an Unnecessarily Deep Horizontal Layer (Layer-skipping Call).....	41
7.1.12	CWE-1064 Invokable Control Element with Signature Containing an Excessive Number of Parameters	41
7.1.13	CWE-1084 Invokable Control Element with Excessive File or Data Access Operations	41
7.1.14	CWE-1051 Initialization with Hard-Coded Network Resource Configuration Data ...	42
7.1.15	CWE-1090 Method Containing Access of a Member Element from Another Class	42
7.1.16	CWE-1074 Class with Excessively Deep Inheritance	42
7.1.17	CWE-1086 Class with Excessive Number of Child Classes.....	43
7.1.18	CWE-1041 Use of Redundant Code (Copy-Paste)	43
7.1.19	CWE-1055 Multiple Inheritance from Concrete Classes.....	43
7.1.20	CWE-1045 Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor	44
7.1.21	CWE-1052 Excessive Use of Hard-Coded Literals in Initialization	44

7.1.22	CWE-1048 Invokable Control Element with Large Number of Outward Calls (Excessive Coupling or Fan-out)	44
7.1.23	CWE-1095 Loop Condition Value Update within the Loop	45
7.1.24	CWE-1085 Invokable Control Element with Excessive Volume of Commented-out Code	45
7.1.25	CWE-1047 Modules with Circular Dependencies	45
7.1.26	CWE-1080 Source Code File with Excessive Number of Lines of Code	46
7.1.27	CWE-1062 Parent Class Element with References to Child Class	46
7.1.28	CWE-1087 Class with Virtual Method without a Virtual Destructor	46
7.1.29	CWE-1079 Parent Class without Virtual Destructor Method	47
7.1.30	Maintainability Detection Patterns	47
7.2	Weakness Category Performance Efficiency	48
7.2.1	CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')	48
7.2.2	Weakness CWE-404 Improper Resource Shutdown or Release	48
7.2.3	CWE-424 Improper Protection of Alternate Path	49
7.2.4	CWE-772 Missing Release of Resource after Effective Lifetime	49
7.2.5	CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime	49
7.2.6	CWE-1073 Non-SQL Invokable Control Element with Excessive Number of Data Resource Access	49
7.2.7	CWE-1057 Data Access Operations Outside of Designated Data Manager Component	50
7.2.8	CWE-1043 Storable and Member Data Element Excessive Number of Aggregated Storable and Member Data Elements	50
7.2.9	CWE-1072 Data Resource Access without use of Connection Pooling	50
7.2.10	CWE-1060 Excessive Number of Inefficient Server-Side Data Accesses	51
7.2.11	CWE-1091 Use of Object without Invoking Destructor Method	51
7.2.12	CWE-1046 Creation of Immutable Text Using String Concatenation	51
7.2.13	CWE-1042 Static Member Data Element outside of a Singleton Class Element	52
7.2.14	CWE-1049 Excessive Data Query Operations in a Large Data Table	52
7.2.15	CWE-1067 Excessive Execution of Sequential Searches of Data Resource	52
7.2.16	CWE-1089 Large Data Table with Excessive Number of Indices	53
7.2.17	CWE-1094 Excessive Index Range Scan for a Data Resource	53
7.2.18	CWE-1050 Excessive Platform Resource Consumption within a Loop	53
7.2.19	CWE-1060 Excessive Number of Inefficient Server-Side Data Accesses	54
7.2.20	Performance Efficiency Detection Patterns	54
7.3	Weakness Category Reliability	54
7.3.1	CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer	54
7.3.2	CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	55
7.3.3	CWE-123 Write-what-where Condition	55
7.3.4	CWE-125 Out-of-bounds Read	56
7.3.5	CWE-130 Improper Handling of Length Parameter Inconsistency	56
7.3.6	CWE-131 Incorrect Calculation of Buffer Size	56
7.3.7	CWE-170 Improper Null Termination	57
7.3.8	CWE-194 Unexpected Sign Extension	57
7.3.9	CWE-195 Signed to Unsigned Conversion Error	57
7.3.10	CWE-196 Unsigned to Signed Conversion Error	58
7.3.11	CWE-197 Numeric Truncation Error	58
7.3.12	CWE-248 Uncaught Exception	58
7.3.13	CWE-252 Unchecked Return Value	59
7.3.14	CWE-366 Race Condition within a Thread	59
7.3.15	CWE-369 Divide by Zero	59

7.3.16	CWE-390 Detection of Error Condition Without Action	59
7.3.17	CWE-391 Unchecked Error Condition	60
7.3.18	CWE-392 Missing Report of Error Condition	60
7.3.19	CWE-394 Unexpected Status Code or Return Value	60
7.3.20	CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')	61
7.3.21	CWE-404 Improper Resource Shutdown or Release	61
7.3.22	CWE-415 Double Free	61
7.3.23	CWE-416 Use After Free	62
7.3.24	CWE-424 Improper Protection of Alternate Path	62
7.3.25	CWE-456 Missing Initialization of a Variable	62
7.3.26	CWE-459 Incomplete Cleanup	63
7.3.27	CWE-476 NULL Pointer Dereference	63
7.3.28	CWE-480 Use of Incorrect Operator	63
7.3.29	CWE-484 Omitted Break Statement in Switch	64
7.3.30	CWE-543 Use of Singleton Pattern Without Synchronization in a Multithreaded Context	64
7.3.31	CWE-562 Return of Stack Variable Address	64
7.3.32	CWE-567 Unsynchronized Access to Shared Data in a Multithreaded Context	64
7.3.33	CWE-595 Comparison of Object References Instead of Object Contents	65
7.3.34	CWE-597 Use of Wrong Operator in String Comparison	65
7.3.35	CWE-662 Improper Synchronization	65
7.3.36	CWE-667 Improper Locking	66
7.3.37	CWE-672 Operation on a Resource after Expiration or Release	67
7.3.38	CWE-681 Incorrect Conversion between Numeric Types	67
7.3.39	CWE-682 Incorrect Calculation	67
7.3.40	CWE-703 Improper Check or Handling of Exceptional Conditions	68
7.3.41	CWE-704 Incorrect Type Conversion or Cast	68
7.3.42	CWE-758 Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	68
7.3.43	CWE-764 Multiple Locks of a Critical Resource	69
7.3.44	CWE-772 Missing Release of Resource after Effective Lifetime	69
7.3.45	CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime	69
7.3.46	CWE-786 Access of Memory Location Before Start of Buffer	70
7.3.47	CWE-787 Out-of-bounds Write	70
7.3.48	CWE-788 Access of Memory Location After End of Buffer	70
7.3.49	CWE-805 Buffer Access with Incorrect Length Value	71
7.3.50	CWE-820 Missing Synchronization	71
7.3.51	CWE-821 Incorrect Synchronization	71
7.3.52	7.3.52 CWE-822 Untrusted Pointer Dereference	72
7.3.53	7.3.53 CWE-823 Use of Out-of-range Pointer Offset	72
7.3.54	CWE-824 Access of Uninitialized Pointer	72
7.3.55	CWE-825 Expired Pointer Dereference	73
7.3.56	CWE-833 Deadlock	73
7.3.57	CWE-835 Loop with Unreachable Exit Condition ('Infinite Loop')	73
7.3.58	CWE-908 Use of Uninitialized Resource	74
7.3.59	CWE-1083 Data Access from Outside Designated Data Manager Component	74
7.3.60	CWE-1058 Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element	74
7.3.61	CWE-1096 Singleton Class Instance Creation without Proper Locking or Synchronization	75
7.3.62	CWE-1087 Class with Virtual Method without a Virtual Destructor	75
7.3.63	CWE-1079 Parent Class without Virtual Destructor Method	75

7.3.64 CWE-1045 Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor76

7.3.65 CWE-1051 Initialization with Hard-Coded Network Resource Configuration Data ...76

7.3.66 CWE-1088 Synchronous Access of Remote Resource without Timeout.....76

7.3.67 CWE-1066 Missing Serialization Control Element77

7.3.68 CWE-1070 Serializable Storable Data Element with non-Serializable Item Elements77

7.3.69 CWE-1097 Persistent Storable Data Element without Associated Comparison Control Element.....77

7.3.70 CWE-1098 Data Element containing Pointer Item without Proper Copy Control Element.....77

7.3.71 CWE-1082 Class Instance Self Destruction Control Element.....78

7.3.72 CWE-1077 Floating Point Comparison with Incorrect Operator78

7.3.73 CWE-665 Improper Initialization.....78

7.3.74 CWE-457 Use of Uninitialized Variable79

7.3.75 Reliability Detection Patterns79

7.4 Weakness Category Security80

7.4.1 Improper Restriction of Operations within the Bounds of a Memory Buffer.....80

7.4.2 CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow').....81

7.4.3 CWE-123 Write-what-where Condition81

7.4.4 CWE-125 Out-of-bounds Read82

7.4.5 CWE-129 Improper Validation of Array Index82

7.4.6 CWE-130 Improper Handling of Length Parameter Inconsistency.....82

7.4.7 CWE-131 Incorrect Calculation of Buffer Size.....83

7.4.8 CWE-134 Use of Externally-Controlled Format String.....83

7.4.9 CWE-194 Unexpected Sign Extension83

7.4.10 CWE-195 Signed to Unsigned Conversion Error83

7.4.11 CWE-196 Unsigned to Signed Conversion Error84

7.4.12 CWE-197 Numeric Truncation Error.....84

7.4.13 CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')84

7.4.14 CWE-23 Relative Path Traversal.....85

7.4.15 CWE-252 Unchecked Return Value.....85

7.4.16 CWE-259 Use of Hard-coded Password.....85

7.4.17 CWE-321 Use of Hard-coded Cryptographic Key86

7.4.18 CWE-36 Absolute Path Traversal86

7.4.19 CWE-366 Race Condition within a Thread86

7.4.20 CWE-369 Divide by Zero87

7.4.21 CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')87

7.4.22 CWE-404 Improper Resource Shutdown or Release87

7.4.23 CWE-424 Improper Protection of Alternate Path.....88

7.4.24 CWE-434 Unrestricted Upload of File with Dangerous Type88

7.4.25 CWE-456 Missing Initialization of a Variable.....88

7.4.26 CWE-457 Use of Uninitialized Variable89

7.4.27 CWE-477 Use of Obsolete Function.....89

7.4.28 CWE-480 Use of Incorrect Operator89

7.4.29 CWE-502 Deserialization of Untrusted Data90

7.4.30 CWE-543 Use of Singleton Pattern Without Synchronization in a Multithreaded Context90

7.4.31 CWE-564 SQL Injection: Hibernate90

7.4.32 CWE-567 Unsynchronized Access to Shared Data in a Multithreaded Context90

7.4.33 CWE-570 Expression is Always False91

7.4.34	CWE-571 Expression is Always True	91
7.4.35	CWE-606 Unchecked Input for Loop Condition	91
7.4.36	CWE-643 Improper Neutralization of Data within XPath Expressions ('XPath Injection')	92
7.4.37	CWE-652 Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')	92
7.4.38	CWE-662 Improper Synchronization	92
7.4.39	CWE-665 Improper Initialization	93
7.4.40	CWE-667 Improper Locking	93
7.4.41	CWE-672 Operation on a Resource after Expiration or Release	94
7.4.42	CWE-681 Incorrect Conversion between Numeric Types	94
7.4.43	CWE-682 Incorrect Calculation	94
7.4.44	CWE-732 Incorrect Permission Assignment for Critical Resource	95
7.4.45	CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection')	95
7.4.46	CWE-772 Missing Release of Resource after Effective Lifetime	95
7.4.47	CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime	96
7.4.48	CWE-778 Insufficient Logging	96
7.4.49	CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	96
7.4.50	CWE-783 Operator Precedence Logic Error	97
7.4.51	CWE-786 Access of Memory Location Before Start of Buffer	97
7.4.52	CWE-787 Out-of-bounds Write	97
7.4.53	CWE-788 Access of Memory Location After End of Buffer	98
7.4.54	CWE-789 Uncontrolled Memory Allocation	98
7.4.55	CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	98
7.4.56	CWE-798 Use of Hard-coded Credentials	98
7.4.57	CWE-805 Buffer Access with Incorrect Length Value	99
7.4.58	CWE-820 Missing Synchronization	99
7.4.59	CWE-821 Incorrect Synchronization	100
7.4.60	CWE-822 Untrusted Pointer Dereference	100
7.4.61	CWE-823 Use of Out-of-range Pointer Offset	100
7.4.62	CWE-824 Access of Uninitialized Pointer	101
7.4.63	CWE-825 Expired Pointer Dereference	101
7.4.64	CWE-835 Loop with Unreachable Exit Condition ('Infinite Loop')	101
7.4.65	CWE-88 Argument Injection or Modification	101
7.4.66	CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	102
7.4.67	CWE-90 Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	102
7.4.68	CWE-91 XML Injection (aka Blind XPath Injection)	102
7.4.69	CWE-99 Improper Control of Resource Identifiers ('Resource Injection')	103
7.4.70	CWE-611 Improper Restriction of XML External Entity Reference ('XXE')	103
7.4.71	CWE-1057 Data Access Control Element from Outside Designated Data Manager Component	103
7.4.72	CWE-415 Double Free	104
7.4.73	CWE-416 Use After Free	104
7.4.74	Security Detection Patterns	104
8	ASCQM Weakness Detection Patterns	107
8.1	Specification of Detection Patterns	107
8.2	ASCQM Check Index of Array Access	107

8.3	ASCQM Check Input of Memory Manipulation Primitives.....	108
8.4	ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities	109
8.5	ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities.....	109
8.6	ASCQM Ban Use of Expired Pointer.....	110
8.7	ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities	111
8.8	ASCQM Check Offset used in Pointer Arithmetic	112
8.9	ASCQM Sanitize User Input used as Pointer	113
8.10	ASCQM Initialize Pointers before Use.....	114
8.11	ASCQM Check NULL Pointer Value before Use.....	115
8.12	ASCQM Ban Use of Expired Resource	116
8.13	ASCQM Ban Double Release of Resource	116
8.14	ASCQM Implement Copy Constructor for Class with Pointer Resource.....	117
8.15	ASCQM Ban Free Operation on Pointer Received as Parameter	118
8.16	ASCQM Ban Delete of VOID Pointer	119
8.17	ASCQM Ban Variable Increment or Decrement Operation in Operations using the Same Variable.....	119
8.18	ASCQM Ban Reading and Writing the Same Variable Used as Assignment Value	120
8.19	ASCQM Handle Return Value of Resource Operations.....	121
8.20	ASCQM Ban Incorrect Numeric Conversion of Return Value.....	122
8.21	ASCQM Handle Return Value of Must Check Operations	123
8.22	ASCQM Check Return Value of Resource Operations Immediately.....	124
8.23	ASCQM Ban Useless Handling of Exceptions.....	125
8.24	ASCQM Ban Incorrect Object Comparison.....	125
8.25	ASCQM Ban Assignment Operation Inside Logic Blocks	126
8.26	ASCQM Ban Comparison Expression Outside Logic Blocks.....	126
8.27	ASCQM Ban Incorrect String Comparison.....	127
8.28	ASCQM Ban Logical Operation with a Constant Operand	128
8.29	ASCQM Implement Correct Object Comparison Operations.....	128
8.30	ASCQM Ban Comma Operator from Delete Statement.....	129
8.31	ASCQM Release in Destructor Memory Allocated in Constructor.....	130
8.32	ASCQM Release Memory after Use with Correct Operation.....	131
8.33	ASCQM Implement Required Operations for Manual Resource Management.....	132
8.34	ASCQM Release Platform Resource after Use.....	133
8.35	ASCQM Release Memory After Use.....	134
8.36	ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor.....	135
8.37	ASCQM Implement Virtual Destructor for Parent Classes.....	135
8.38	ASCQM Release File Resource after Use in Operation.....	136
8.39	ASCQM Implement Virtual Destructor for Classes with Virtual Methods.....	137
8.40	ASCQM Ban Self Destruction.....	138
8.41	ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls.....	138
8.42	ASCQM Ban Non-Final Static Data in Multi-Threaded Context	139
8.43	ASCQM Ban Non-Serializable Elements in Serializable Objects	140
8.44	ASCQM Ban Hard-Coded Literals used to Connect to Resource.....	141
8.45	ASCQM Ban Unintended Paths.....	142
8.46	ASCQM Ban Incorrect Float Number Comparison	143
8.47	ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context.....	143
8.48	ASCQM Ban Incorrect Numeric Implicit Conversion	145
8.49	ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context .	146
8.50	ASCQM Ban Incorrect Synchronization Mechanisms	147

8.51	ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context	148
8.52	ASCQM Ban Incorrect Type Conversion	149
8.53	ASCQM Ban Return of Local Variable Address	150
8.54	ASCQM Ban Storage of Local Variable Address in Global Variable	151
8.55	ASCQM Ban While TRUE Loop Without Path to Break	151
8.56	ASCQM Ban Unmodified Loop Variable Within Loop	152
8.57	ASCQM Check and Handle ZERO Value before Use as Divisor	153
8.58	ASCQM Ban Creation of Lock on Private Non-Static Object to Access Private Static Data	154
8.59	ASCQM Release Lock After Use	154
8.60	ASCQM Ban Sleep Between Lock Acquisition and Release	155
8.61	ASCQM Ban Creation of Lock on Non-Final Object	157
8.62	ASCQM Ban Creation of Lock on Inappropriate Object Type	157
8.63	ASCQM NULL Terminate Output of String Manipulation Primitives	158
8.64	ASCQM Release File Resource after Use in Class	159
8.65	ASCQM Use Break in Switch Statement	160
8.66	ASCQM Catch Exceptions	161
8.67	ASCQM Ban Empty Exception Block	162
8.68	ASCQM Initialize Resource before Use	162
8.69	ASCQM Ban Incompatible Lock Acquisition Sequences	164
8.70	ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues	164
8.71	ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation	165
8.72	ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size	166
8.73	ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value	167
8.74	ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock	168
8.75	ASCQM Initialize Variables	169
8.76	ASCQM Ban Allocation of Memory with Null Size	170
8.77	ASCQM Ban Double Free on Pointers	170
8.78	ASCQM Initialize Variables before Use	171
8.79	ASCQM Ban Self Assignment	172
8.80	ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use	173
8.81	ASCQM Ban Not Operator on Operand of Bitwise Operation	174
8.82	ASCQM Ban Not Operator on Non-Boolean Operand of Comparison Operation	174
8.83	ASCQM Ban Incorrect Joint Comparison	175
8.84	ASCQM Secure XML Parsing with Secure Options	176
8.85	ASCQM Secure Use of Unsafe XML Processing with Secure Parser	178
8.86	ASCQM Sanitize User Input used in Path Manipulation	179
8.87	ASCQM Sanitize User Input used in SQL Access	180
8.88	ASCQM Sanitize User Input used in Document Manipulation Expression	181
8.89	ASCQM Sanitize User Input used in Document Navigation Expression	183
8.90	ASCQM Sanitize User Input used to access Directory Resources	184
8.91	ASCQM Sanitize Stored Input used in User Output	185
8.92	ASCQM Sanitize User Input used in User Output	187
8.93	ASCQM Sanitize User Input used in System Command	188
8.94	ASCQM Ban Use of Deprecated Libraries	189
8.95	ASCQM Sanitize User Input used as Array Index	190
8.96	ASCQM Check Input of Memory Allocation Primitives	191
8.97	ASCQM Sanitize User Input used as String Format	192
8.98	ASCQM Sanitize User Input used in Loop Condition	193
8.99	ASCQM Sanitize User Input used as Serialized Object	195
8.100	ASCQM Log Caught Security Exceptions	196
8.101	ASCQM Ban File Creation with Default Permissions	197

8.102	ASCQM Ban Use of Prohibited Low-Level Resource Management Functionality	198
8.103	ASCQM Ban Excessive Size of Index on Columns of Large Tables	200
8.104	ASCQM Implement Index Required by Query on Large Tables	201
8.105	ASCQM Ban Excessive Number of Index on Columns of Large Tables.....	202
8.106	ASCQM Ban Excessive Complexity of Data Resource Access.....	202
8.107	ASCQM Ban Expensive Operations in Loops.....	203
8.108	ASCQM Limit Number of Aggregated Non-Primitive Data Types.....	205
8.109	ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure.....	206
8.110	ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code	207
8.111	ASCQM Ban Incremental Creation of Immutable Data	208
8.112	ASCQM Ban Static Non-Final Data Element Outside Singleton.....	208
8.113	ASCQM Ban Conversion References to Child Class.....	209
8.114	ASCQM Ban Circular Dependencies between Modules.....	210
8.115	ASCQM Limit Volume of Commented-Out Code.....	211
8.116	ASCQM Limit Size of Operations Code.....	212
8.117	ASCQM Limit Volume of Similar Code	212
8.118	ASCQM Limit Algorithmic Complexity via Cyclomatic Complexity Value	213
8.119	ASCQM Limit Number of Data Access	214
8.120	ASCQM Ban Excessive Number of Children.....	215
8.121	ASCQM Ban Excessive Number of Inheritance Levels	216
8.122	ASCQM Ban Usage of Data Elements from Other Classes	216
8.123	ASCQM Ban Control Flow Transfer	217
8.124	ASCQM Ban Loop Value Update within Incremental and Decremental Loop	218
8.125	ASCQM Limit Number of Parameters	219
8.126	ASCQM Ban Unreferenced Dead Code.....	220
8.127	ASCQM Ban Excessive Number of Concrete Implementations to Inherit From	220
8.128	ASCQM Limit Number of Outward Calls	221
8.129	ASCQM Sanitize User Input used in Expression Language Statement.....	221
8.130	ASCQM Ban Hard-Coded Literals used to Initialize Variables.....	223
8.131	ASCQM Ban Logical Dead Code	223
8.132	ASCQM Ban Exception Definition without Ever Throwing It.....	224
8.133	ASCQM Ban Switch in Switch Statement	225
8.134	ASCQM Limit Algorithmic Complexity via Module Design Complexity Value.....	226
8.135	ASCQM Limit Algorithmic Complexity via Essential Complexity Value	227
8.136	ASCQM Use Default Case in Switch Statement.....	228
9	Calculation of the Quality Measures	229
	Annex A Consortium for IT Software Quality (CISQ)	230
	Annex B Common Weakness Enumeration (CWE)	231
	Annex C Related Quality Measures.....	232
	Annex D Relationship to ISO/IEC 25000 Series Standards.....	233
	References.....	235

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see <http://patents.iec.ch>).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by the Object Management Group (OMG) (as Automated Source Code Quality Measures [ASCQM], Version 1.0) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Computers are being used in an increasing variety of application areas where their correct operation is critical for business accuracy or human safety. Developing or selecting trustworthy, dependable software products is of prime importance. Quantitative evaluation of software product quality is key to ensuring correct performance. This assurance requires rigorous measures of software product quality characteristics. Defining these measures in an international standard is important for standardizing the use of such measures by public and commercial organizations.

ISO/IEC 5055:2020 Automated Source Code Quality Measures defines measures at the software product level (source code) for four of the quality characteristics defined in the software product quality model presented in ISO/IEC 25010-2. These measures conform to the definitions of quality characteristics in ISO/IEC 25010-2 that they quantify. Each measure is calculated from counts of severe weaknesses in the source code that affect the quality characteristic being measured. Each weakness is associated with one or more detection patterns that can guide the development of automated tools for quality analysis of software products. The weaknesses included in each measure were selected by an international team of software engineering experts based on the severity of their impact on a software product quality characteristic.

ISO/IEC 25023 defines software measures for the quality model in ISO/IEC 25010-2, but most all of these measures are at the behavioral level. For instance, a measure for availability is defined as mean downtime, which does not identify the problems in the software that caused the downtime. ISO/IEC 5055:2020 therefore supplements ISO/IEC 25023 by defining software measures at the level of weaknesses in the source code. Thus, availability would be measured by the existence of software weaknesses that cause downtime such as poor error handling or missing timeouts. ISO/IEC 5055:2020 adds strong product level measurement to support the ISO/IEC 25000 software product quality standards.

ISO/IEC 5055:2020 may be revised from time to time based on new severe weaknesses being added or existing weaknesses in the model becoming less severe because of advances in computer science. Thus, this will be a standard that adapts to changes in the technology of computing.

Information technology — Software measurement — Software quality measurement — Automated source code quality measures

1 Scope

1.1 Purpose

The measures in this standard were calculated from detecting and counting violations of good architectural and coding practices in the source code that could result in unacceptable operational risks or excessive costs. Establishing standards for these measures at the source code level is important because they have been used in outsourcing and system development contracts without having international standards to reference. For instance, the ISO/IEC 25000 series of standards that govern software product quality provide only a small set of measures at the source code level.

A primary objective of updating these measures was to extend their applicability to embedded software, which is especially important for the growing implementation of embedded devices and the Internet of Things. Functionality that has traditionally been implemented in IT applications is now being moved to embedded chips. Since the weaknesses included in the measures specified in this document have been found to be applicable to all forms of software, embedded software is not treated separately in this specification.

1.2 Overview of Structural Quality Measurement in Software

Measurement of the structural quality characteristics of software has a long history in software engineering (Curtis, 1980). These characteristics are also referred to as the structural, internal, technical, or engineering characteristics of software source code. Software quality characteristics are increasingly incorporated into development and outsourcing contracts as the equivalent of service level agreements. That is, target thresholds based on structural quality measures are being written into contracts as acceptance criteria for delivered software. Currently there are no standards for most of the software structural quality measures used in contracts. ISO/IEC 25023 purports to address these measures, but most of them are measures of external behavior and do not sufficiently define measures that can be developed from source code during development. Consequently, providers are subject to different interpretations and calculations of common structural quality characteristics in each contract. This specification addresses one aspect of this problem by providing a specification for measuring four structural quality characteristics from the source code—Reliability, Security, Performance Efficiency, and Maintainability.

Recent advances in measuring the structural quality of software involve detecting violations of good architectural and coding practice from statically analyzing source code. Violations of good architectural and design practice can also be detected from statically analyzing design specifications written in a design language with a formal syntax and semantics. Good architectural and coding practices can be stated as rules for engineering software products. Violations of these rules will be called weaknesses in this specification to be consistent with terms used in the Common Weakness Enumeration (Martin & Barnum, 2006) which lists many of the weaknesses used in several of these measures.

The Automated Source Code Quality Measures are correlated measures rather than absolute measures. That is, since they do not measure all possible weaknesses in each of the four areas, they do not provide absolute measures. However, since they include counts of what industry experts have determined to be most severe weaknesses, they provide strong indicators of the quality of a software system in each area. In most instances they will be highly correlated with the probability of operational or cost problems related to each measure's area.

Recent research in analyzing structural quality weaknesses has identified common patterns of code structures that can be used to detect weaknesses. Many of these 'Detection Patterns' are shared across different weaknesses. Detection Patterns will be used in this specification to organize and simplify the presentation of weaknesses underlying the four structural quality measures. Each weakness will be described as a quality measure element to remain consistent with ISO/IEC 25020. Each quality measure element will be represented as one or more Detection Patterns. Many quality measure elements (weaknesses) will share one or more Detection Patterns in common.

The normative portion of this specification represents each quality attribute (weakness) and quality measure element (detection pattern) using the Structured Patterns Metamodel Standard (SPMS). The code-based elements in these patterns are represented using the Knowledge Discovery Metamodel (KDM). The score for each of the four Automated Source Code Quality Measures from their quality measure elements is calculated by counting the number of detection patterns for each weakness, and then summing these numbers for all the weaknesses included in the specific quality characteristic measure.

2 Conformance

Implementations of this specification should be able to demonstrate the following attributes to claim conformance—automated, objective, transparent, and verifiable.

- **Automated**—The analysis of the source code and counting of weaknesses shall be fully automated. The initial inputs required to prepare the source code for analysis include the source code of the application, the artifacts and information needed to configure the application for operation, and any available description of the architectural layers in the application.
- **Objective**—After the source code has been prepared for analysis using the information provided as inputs, the analysis, calculation, and presentation of results shall not require further human intervention. The analysis and calculation shall be able to repeatedly produce the same results and outputs on the same body of software.
- **Transparent**—Implementations that conform to this specification shall clearly list all source code (including versions), non-source code artifacts, and other information used to prepare the source code for submission to the analysis.
- **Verifiable**—Compliance with this specification requires that an implementation shall state the assumptions/heuristics it uses with sufficient detail so that the calculations may be independently verified by third parties. In addition, all inputs used shall be clearly described and itemized so that they can be audited by a third party.

3 Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this specification. Dated references, subsequent amendments to, or revisions of any of these publications do not apply.

- Structured Patterns Metamodel Standard, formal/2017-11-01, <https://www.omg.org/spec/SPMS/1.2/>
- ISO/IEC 19506:2012 – Object Management Group Architecture Driven Modernization (ADM) – Knowledge Discovery Metamodel (KDM). Also, Knowledge Discovery Metamodel, version 1.4 (KDM), formal/2016-09-01, <https://www.omg.org/spec/KDM/1.4/>
- MOF/XMI Mapping, version 2.5.1 (XMI), <https://www.omg.org/spec/XMI/2.5.1/>
- ISO/IEC 25010:2011 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models
- ISO/IEC 25020:2019 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Measurement reference model and guide
- ISO/IEC 19515:2019, Automated Function Points. Information technology -- Object Management Group Automated Function Points (AFP), 1.0. Geneva, Switzerland. Also, Object Management Group (2014). Automated Function Points - formal/2014-01-03 <https://www.omg.org/spec/AFP/>. Needham, MA: Object Management Group.
- ITU-T X.1524 – Series X: Data Networks, Open System Communications and Security – Cybersecurity information exchange – Vulnerability/state exchange – Common weakness enumeration.