DRAFT International Standard

ISO/IEC DIS 5055-2

Information technology — Automated source code quality measures —

Part 2: Data protection measure

ICS: 35.080

ISO/IEC JTC **1**

Secretariat: **ANSI**

Voting begins on:
2025-12-02

Voting terminates on:
2026-02-24

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENTS AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

This document is circulated as received from the committee secretariat.

Reference number
ISO/IEC DIS 5055-2:2025(en)

© ISO/IEC 2025



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2025

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

Table of Contents

1 Scope 1
1.1 Purpose 1
1.2 Overview of Structural Quality Measurement in Software 1
2 Conformance 2
3 Normative References 2
4 Terms and Definitions 3
5 Symbols (and Abbreviated Terms) 4
6 Additional Information (Informative) 5
6.1 Software Product Inputs 5
6.2 Automated Source Code Data Protection Measure Elements 5
6.3 Specification of Data Protection Measure Elements 10
6.4 Specification of Detection Patterns 10
6.5 Knowledge Discovery Metamodel (KDM) 11
6.6 Software Patterns Metamodel Standard (SPMS) 15
6.7 Reading guide 16
7 List of ASCDPM Weaknesses (Normative) 17
7.1 CWE-22 — Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') 17
7.2 CWE-23 — Relative Path Traversal 17
7.3 CWE-36 — Absolute Path Traversal 17
7.4 CWE-77 — Improper Neutralization of Special Elements used in a Command ('Command Injection') 18
7.5 CWE-78 — Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') 18
7.6 CWE-79 — Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') 19
7.7 CWE-88 — Argument Injection or Modification 19
7.8 CWE-89 — Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') 19
7.9 CWE-90 — Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection') 20
7.10 CWE-91 — XML Injection (aka Blind XPath Injection) 20
7.11 CWE-99 — Improper Control of Resource Identifiers ('Resource Injection') 20
7.12 CWE-119 — Improper Restriction of Operations within the Bounds of a Memory Buffer 20
7.13 CWE-120 — Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') 21
7.14 CWE-123 — Write-what-where Condition 22
7.15 CWE-125 — Out-of-bounds Read 22
7.16 CWE-129 — Improper Validation of Array Index 22
7.17 CWE-130 — Improper Handling of Length Parameter Inconsistency 22
7.18 CWE-131 — Incorrect Calculation of Buffer Size 23
7.19 CWE-134 — Use of Externally-Controlled Format String 23
7.20 CWE-170 — Improper Null Termination 23
7.21 CWE-194 — Unexpected Sign Extension 24
7.22 CWE-195 — Signed to Unsigned Conversion Error 24
7.23 CWE-196 — Unsigned to Signed Conversion Error 24
7.24 CWE-197 — Numeric Truncation Error 25
7.25 CWE-213 — Exposure of Sensitive Information Due to Incompatible Policies 25
7.26 CWE-248 — Uncaught Exception 25
7.27 CWE-259 — Use of Hard-coded Password 26
7.28 CWE-284 — Improper Access Control 26
7.29 CWE-285 — Improper Authorization 26
7.30 CWE-287 — Improper Authentication 27
7.31 CWE-288 — Authentication Bypass Using an Alternate Path or Channel 27
7.32 CWE-311 — Missing Encryption of Sensitive Data 28

© ISO/IEC 2025 – All rights reserved

7.33 CWE-321 — Use of Hard-coded Cryptographic Key 28
7.34 CWE-359 — Exposure of Private Personal Information to an Unauthorized Actor 28
7.35 CWE-366 — Race Condition within a Thread 28
7.36 CWE-369 — Divide By Zero 29
7.37 CWE-391 — Unchecked Error Condition 29
7.38 CWE-392 — Missing Report of Error Condition 29
7.39 CWE-404 — Improper Resource Shutdown or Release 30
7.40 CWE-415 — Double Free 30
7.41 CWE-416 — Use After Free 31
7.42 CWE-424 — Improper Protection of Alternate Path 31
7.43 CWE-434 — Unrestricted Upload of File with Dangerous Type 31
7.44 CWE-456 — Missing Initialization of a Variable 32
7.45 CWE-457 — Use of Uninitialized Variable 32
7.46 CWE-502 — Deserialization of Untrusted Data 32
7.47 CWE-543 — Use of Singleton Pattern Without Synchronization in a Multithreaded Context 33
7.48 CWE-562 — Return of Stack Variable Address 33
7.49 CWE-567 — Unsynchronized Access to Shared Data in a Multithreaded Context 33
7.50 CWE-606 — Unchecked Input for Loop Condition 34
7.51 CWE-611 — Improper Restriction of XML External Entity Reference ('XXE') 34
7.52 CWE-624 — Executable Regular Expression Error 34
7.53 CWE-639 — Authorization Bypass Through User-Controlled Key 34
7.54 CWE-643 — Improper Neutralization of Data within XPath Expressions ('XPath Injection') 35
7.55 CWE-652 — Improper Neutralization of Data within XQuery Expressions ('XQuery Injection') 35
7.56 CWE-662 — Improper Synchronization 35
7.57 CWE-665 — Improper Initialization 36
7.58 CWE-667 — Improper Locking 37
7.59 CWE-672 — Operation on a Resource after Expiration or Release 37
7.60 CWE-681 — Incorrect Conversion between Numeric Types 38
7.61 CWE-682 — Incorrect Calculation 38
7.62 CWE-703 — Improper Check or Handling of Exceptional Conditions 38
7.63 CWE-704 — Incorrect Type Conversion or Cast 39
7.64 CWE-732 — Incorrect Permission Assignment for Critical Resource 39
7.65 CWE-761 — Free of Pointer not at Start of Buffer 39
7.66 CWE-762 — Mismatched Memory Management Routines 40
7.67 CWE-763 — Release of Invalid Pointer or Reference 40
7.68 CWE-764 — Multiple Locks of a Critical Resource 40
7.69 CWE-772 — Missing Release of Resource after Effective Lifetime 41
7.70 CWE-775 — Missing Release of File Descriptor or Handle after Effective Lifetime 41
7.71 CWE-786 — Access of Memory Location Before Start of Buffer 41
7.72 CWE-787 — Out-of-bounds Write 42
7.73 CWE-788 — Access of Memory Location After End of Buffer 42
7.74 CWE-798 — Use of Hard-coded Credentials 42
7.75 CWE-805 — Buffer Access with Incorrect Length Value 43
7.76 CWE-820 — Missing Synchronization 43
7.77 CWE-821 — Incorrect Synchronization 43
7.78 CWE-822 — Untrusted Pointer Dereference 44
7.79 CWE-823 — Use of Out-of-range Pointer Offset 44
7.80 CWE-824 — Access of Uninitialized Pointer 44
7.81 CWE-825 — Expired Pointer Dereference 45
7.82 CWE-862 — Missing Authorization 45
7.83 CWE-863 — Incorrect Authorization 45
7.84 CWE-908 — Use of Uninitialized Resource 46
7.85 CWE-915 — Improperly Controlled Modification of Dynamically-Determined Object Attributes 46

© ISO/IEC 2025 – All rights reserved

7.86 CWE-917 — Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection') 46
7.87 CWE-1051 — Storable and Member Data Element Initialization with Hard-Coded Network Resource Configuration Data 47
7.88 CWE-1058 — Named Callable and Method Control Element in Multi-Thread Context with non-Final Static Storable or Member Element 47
7.89 CWE-1096 Singleton Class Instance Creation without Proper Lock Element Management 47

8 ASCQM Weakness Detection Patterns (Normative) 49
8.1 ASCQM Check Index of Array Access 49
8.2 ASCQM Check Input of Memory Manipulation Primitives 50
8.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities 50
8.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities 51
8.5 ASCQM Ban Use of Expired Pointer 52
8.6 ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities 53
8.7 ASCQM Check Offset used in Pointer Arithmetic 53
8.8 ASCQM Sanitize User Input used as Pointer 54
8.9 ASCQM Initialize Pointers before Use 55
8.10 ASCQM Ban Use of Expired Resource 56
8.11 ASCQM Ban Double Release of Resource 57
8.12 ASCQM Implement Copy Constructor for Class with Pointer Resource 57
8.13 ASCQM Ban Free Operation on Pointer Received as Parameter 58
8.14 ASCQM Ban Useless Handling of Exceptions 59
8.15 ASCQM Ban Comma Operator from Delete Statement 59
8.16 ASCQM Release in Destructor Memory Allocated in Constructor 60
8.17 ASCQM Release Memory after Use with Correct Operation 61
8.18 ASCQM Implement Required Operations for Manual Resource Management 62
8.19 ASCQM Release Platform Resource after Use 63
8.20 ASCQM Release Memory After Use 64
8.21 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor 65
8.22 ASCQM Implement Virtual Destructor for Parent Classes 66
8.23 ASCQM Release File Resource after Use in Operation 66
8.24 ASCQM Implement Virtual Destructor for Classes with Virtual Methods 67
8.25 ASCQM Ban Non-Final Static Data in Multi-Threaded Context 68
8.26 ASCQM Ban Hard-Coded Literals used to Connect to Resource 68
8.27 ASCQM Ban Unintended Paths 69
8.28 ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context 70
8.29 ASCQM Ban Incorrect Numeric Implicit Conversion 71
8.30 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context 72
8.31 ASCQM Ban Incorrect Synchronization Mechanisms 73
8.32 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context 74
8.33 ASCQM Ban Incorrect Type Conversion 75
8.34 ASCQM Ban Return of Local Variable Address 76
8.35 ASCQM Ban Storage of Local Variable Address in Global Variable 76
8.36 ASCQM Check and Handle ZERO Value before Use as Divisor 77
8.37 ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data 78
8.38 ASCQM Release Lock After Use 79
8.39 ASCQM Ban Sleep Between Lock Acquisition and Release 79
8.40 ASCQM Ban Creation of Lock On Non-Final Object 80
8.41 ASCQM Ban Creation of Lock On Inappropriate Object Type 81
8.42 ASCQM NULL Terminate Output of String Manipulation Primitives 82
8.43 ASCQM Release File Resource after Use in Class 83
8.44 ASCQM Catch Exceptions 83
8.45 ASCQM Ban Empty Exception Block 84
8.46 ASCQM Initialize Resource before Use 85
8.47 ASCQM Ban Incompatible Lock Acquisition Sequences 86

© ISO/IEC 2025 – All rights reserved

8.48 ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation 87
8.49 ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size 87
8.50 ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value 88
8.51 ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock 89
8.52 ASCQM Initialize Variables 90
8.53 ASCQM Ban Allocation of Memory with Null Size 91
8.54 ASCQM Ban Double Free On Pointers 92
8.55 ASCQM Initialize Variables before Use 93
8.56 ASCQM Ban Self Assignment 93
8.57 ASCQM Secure XML Parsing with Secure Options 94
8.58 ASCQM Secure Use of Unsafe XML Processing with Secure Parser 95
8.59 ASCQM Sanitize User Input used in Path Manipulation 96
8.60 ASCQM Sanitize User Input used in SQL Access 97
8.61 ASCQM Sanitize User Input used in Document Manipulation Expression 98
8.62 ASCQM Sanitize User Input used in Document Navigation Expression 100
8.63 ASCQM Sanitize User Input used to access Directory Resources 101
8.64 ASCQM Sanitize Stored Input used in User Output 102
8.65 ASCQM Sanitize User Input used in User Output 103
8.66 ASCQM Sanitize User Input used in System Command 105
8.67 ASCQM Sanitize User Input used as Array Index 106
8.68 ASCQM Sanitize User Input used as String Format 107
8.69 ASCQM Sanitize User Input used in Loop Condition 108
8.70 ASCQM Sanitize User Input used as Serialized Object 110
8.71 ASCQM Ban File Creation with Default Permissions 111
8.72 ASCQM Ban Unintended Paths Bypassing Authentication 112
8.73 ASCQM Ban Unintended Paths Bypassing Authorization 113
8.74 ASCQM Ban Unintented Paths To Sensitive Data 114
8.75 ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues 116
8.76 ASCQM Catch Authentication Exceptions 116
8.77 ASCQM Catch Authorization Exceptions 117
8.78 ASCQM Check Return Value of Authentication Operations Immediately 118
8.79 ASCQM Check Return Value of Authorization Operations Immediately 119
8.80 ASCQM Encrypt User Input used in SQL Access to Sensitive Data 119
8.81 ASCQM Release Memory after Use with Correct Reference 121
8.82 ASCQM Sanitize Deserialized Object used in Stored Data 122
8.83 ASCQM Sanitize User Input used in Expression Language Statement 123
8.84 ASCQM Sanitize User Input used in SQL Access to primary keys 124
8.85 ASCQM Sanitize User Input used in URI Building 125

9 Calculation of Quality and Functional Density Measures (Normative) 128
9.1 Calculation of the Base Measure 128
9.2 Functional Density of Weaknesses 128

10 References (Informative) 130

Annex A: Consortium for IT Software Quality (CISQ) (Informative) 131

Annex B: Common Weakness Enumeration (CWE) (Informative) 132

Annex C: Comparison of Weaknesses Included in the Automated Source Code Security, Reliability, and Data Protection Measures (Informative) 133

Annex D: Relationship of the Automated Source Code Data Protection Measure to ISO 25000 Series Standards (SQuaRE) (Informative) 136

© ISO/IEC 2025 – All rights reserved

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 5055-1:2021 was prepared by the Consortium for Information and Software Quality, a special interest group managed by the Object Management Group chartered to produce standards for automating the measurement of software size and quality.

© ISO/IEC 2025 – All rights reserved

© ISO/IEC 2025 – All rights reserved

1 Scope

1.1 Purpose

This is the second standard in the multipart Automated Source Code Quality Measures (ASCQM) standard (ISO/IEC 5055-1:2021). It covers common weaknesses (CWEs) that affect the protection of confidential information. Specifying this measure is important as a source of evidence for complying with regulations such as the General Data Protection Regulation (GDPR) in Europe, and in the United States the Cybersecurity Maturity Model Certification (CMMC), California Consumer Privacy Act, the California Consumer Privacy Act enhanced by the California Privacy Rights Act (CPRA), the Health Insurance Portability and Accountability Act (HIPAA) enhanced with the Health Information Technology for Economic and Clinical Health (HITECH) Act, and the Gramm-Leach-Bliley Act (GLBA) for financial services.

This measure is calculated from detecting and counting 89 violations of good architectural and coding practices (weaknesses) in the source code that could result in unacceptable risks to the exposure or theft of confidential information. This measure will supplement ISO/IEC 25023 that provides measures of software product confidentiality (a subcharacteristic of Security) by providing a measure at the source code level for protecting confidential data.

1.2 Overview of Structural Quality Measurement in Software

Many recent Governmental regulations are requiring evidence that software-intensive systems provide protection of confidential information. Much of the evidence provided involves the process by which these systems are developed and accessed. However, these regulations are often weak on the evidence required to indicate the systems themselves are secure. This specification addresses one aspect of this problem by providing measure of the extent to which a software system is free from weaknesses that would expose confidential information to unauthorized parties. Thus, this specification provides a measure calculated from detecting weaknesses affecting data protection in the source code.

Measurement of the structural quality characteristics of software such as data protection has a long history in software engineering (Curtis, 1980). Recent advances in measuring the structural quality of software involve detecting violations of good architectural and coding practice from statically analyzing source code. 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 includes weaknesses that affect data protection.

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 data protection. 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 calculation of the Automated Source Code Data Protection Measure from their quality measure elements is then represented in the Structured Metrics Metamodel (SMM). This calculation is developed 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.

© ISO/IEC 2025 – All rights reserved

2 Conformance

Implementations of this specification shall demonstrate the following attributes to claim conformance: automated, objective, transparent, and verifiable.

- ***Automated***—The analysis of the source code and counting of weaknesses must 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 must not require further human intervention. The analysis and calculation must be able to repeatedly produce the same results and outputs on the same body of software.
- ***Transparent***—Implementations that conform to this specification must 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 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 are required to 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, https://www.omg.org/spec/SPMS/1.2/
- Knowledge Discovery Metamodel, version 1.4 (KDM), https://www.omg.org/spec/KDM/1.4/
- Structured Metrics Metamodel, version 1.2 (SMM), formal/2012-01-05, https://www.omg.org/spec/SMM/1.2/PDF
- MOF/XMI Mapping, version 2.5.1 (XMI), https://www.omg.org/spec/XMI/2.5.1/
- International Organization for Standards (2007). *ISO/IEC 25020:2007 Software engineering — Software measurement — Functional size measurement —Part 1: Definition of concepts*
- International Organization for Standards (2019). *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

© ISO/IEC 2025 – All rights reserved

4 Terms and Definitions

For the purposes of this specification, the following terms and definitions apply.

Automated Function Points—a specification for automating the counting of Function Points that mirrors as closely as possible the counting guidelines of the International Function Point User Group. (OMG, formal 2014-01-03)

Common Weakness Enumeration—a repository maintained by MITRE Corporation of known weaknesses in software that can be exploited to gain unauthorized entry into a software system. (cwe.mitre.org)

Contributing Weakness—a weakness that is represented as a child of a parent weakness in the Common Weakness Enumeration, that is, a variant instantiation of the parent weakness (cwe.mitre.org)

Data Protection—the ability of a software product to prevent unauthorized access to confidential information contained within the product or within any software product it interacts with.

Detection Pattern—a collection of parsed program elements and their relations that constitute a weakness in the software.

Parent Weakness—a weakness in the Common Weakness Enumeration that has numerous possible instantiations in software that are represented by its relation to child CWEs (cwe.mitre.org)

Data Protection Measure Element—a measure defined in terms of a software quality attribute and the measurement method for quantifying it, including optionally the transformation by a mathematical function (adapted from ISO/IEC 25020)

Security— capability of a product to protect information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization, and to defend against attack patterns by malicious actors (ISO/IEC 25010)

Software Product—a set of computer programs, procedures, and possibly associated documentation and data. (ISO/IEC 25010)

Software Data Protection Attribute—an inherent property or characteristic of software that can be distinguished quantitatively or qualitatively by human or automated means. (adapted from ISO/IEC 25020)

Software Data Protection Rule—an architectural or coding practice or convention that represents good software engineering practice and avoids problems in software development, maintenance, or operations. Violations of these quality rules produces software anti-patterns.

Structural Element—a component of software code that can be uniquely identified and counted such as a token, decision, variable, etc.

Weakness— sometimes referred to as a software anti-pattern, is a pattern or structure in the code (Detection Pattern) that is inconsistent with good architectural or coding practice, violates a software quality rule, and can lead to operational or cost problems. (derived from cwe.mitre.com)

© ISO/IEC 2025 – All rights reserved

5 Symbols (and Abbreviated Terms)

AFP — Automated Function Points

ASCDPM — Automated Source Code Daa Protection Measure

ASCQM — Automated Source Code Quality Measures

ASCSM — Automated Source Code Security Measure

CWE — Common Weakness Enumeration

CISQ — Consortium for Information and Software Quality

KDM — Knowledge Discovery Metamodel

SPMS — Structured Pattern Metamodel Standard

SMM — Structured Metrics Metamodel

© ISO/IEC 2025 – All rights reserved