DRAFT International Standard

ISO/IEC DIS 5055-3

Information technology — Automated source code quality measures —

Part 3: Resource sustainability 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 5055-3:2025(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 3
3 Normative References 3
4 Terms and Definitions 4
5 Symbols (and Abbreviated Terms) 6
6 Additional Information (Informative) 7
6.1 Software Product Inputs 7
6.2 Automated Source Code Quality Measure Elements 7
6.3 Automated Source Code Resource Sustainability Measure Element Descriptions 7
6.4 Introduction to the Specification of Quality Measure Elements 12
6.5 Knowledge Discovery Metamodel (KDM) 12
6.6 Software Patterns Metamodel Standard (SPMS) 14
6.7 Specification of Detection Patterns 15
6.8 Reading guide 16
7 ASCRSM Weakness Specifications (Normative) 17
7.1 CWE-248 Uncaught Exception 17
7.2 CWE-252 Unchecked Return Value 17
7.3 CWE-390 Detection of Error Condition Without Action 17
7.4 CWE-391 Unchecked Error Condition 17
7.5 CWE-392 Missing Report of Error Condition 18
7.6 CWE-394 Unexpected Status Code or Return Value 18
7.7 CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak') 18
7.8 CWE-404 Improper Resource Shutdown or Release 19
7.9 CWE-424 Improper Protection of Alternate Path 19
7.10 CWE-459 Incomplete Cleanup 19
7.11 CWE-703 Improper Check or Handling of Exceptional Conditions 20
7.12 CWE-762 Mismatched Memory Management Routines 20
7.13 CWE-772 Missing Release of Resource after Effective Lifetime 20
7.14 CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime 21
7.15 CWE-833 Deadlock 21
7.16 CWE-835 Loop with Unreachable Exit Condition ('Infinite Loop') 21
7.17 CWE-1043 Storable and Member Data Element Excessive Number of Aggregated Storable and Member Data Elements 22
7.18 CWE-1046 Creation of Immutable Text Using String Concatenation 22
7.19 CWE-1049 Excessive Data Query Operations in a Large Data Table 22
7.20 CWE-1050 Excessive Platform Resource Consumption within a Loop 23
7.21 CWE-1051 Initialization with Hard-Coded Network Resource Configuration Data 23
7.22 CWE-1057 Data Access Operations Outside of Designated Data Manager Component 23
7.23 CWE-1060 Excessive Number of Inefficient Server-Side Data Accesses 24
7.24 CWE-1067 Excessive Execution of Sequential Searches of Data Resource 24
7.25 CWE-1069 Empty Exception Block 24

© ISO/IEC 2025 – All rights reserved

7.26 CWE-1072 Data Resource Access without use of Connection Pooling 25
7.27 CWE-1073 Non-SQL Invokable Control Element with Excessive Number of Data Resource Access 25
7.28 CWE-1083 Data Access from Outside Designated Data Manager Component 25
7.29 CWE-1088 Synchronous Access of Remote Resource without Timeout 26
7.30 CWE-1089 Large Data Table with Excessive Number of Indices 26
7.31 CWE-1091 Use of Object without Invoking Destructor Method 26
7.32 CWE-1094 Excessive Index Range Scan for a Data Resource 26
7.33 CWE-1235 Incorrect Use of Autoboxing and Unboxing for Performance Critical Operations 27
7.34 ASCRSM Detection Patterns 27

8 ASCRSM Weakness Detection Patterns (Normative) 29
8.1 ASCQM Ban Incorrect Numeric Conversion of Return Value 29
8.2 ASCQM Handle Return Value of Must Check Operations 29
8.3 ASCQM Handle Return Value of Resource Operations 30
8.4 ASCQM Check Return Value of Resource Operations Immediately 32
8.5 ASCQM Ban Useless Handling of Exceptions 32
8.6 ASCQM Ban Comma Operator from Delete Statement 33
8.7 ASCQM Release in Destructor Memory Allocated in Constructor 34
8.8 ASCQM Release Memory after Use with Correct Operation 35
8.9 ASCQM Implement Required Operations for Manual Resource Management 36
8.10 ASCQM Release Platform Resource after Use 37
8.11 ASCQM Release Memory After Use 38
8.12 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor 39
8.13 ASCQM Implement Virtual Destructor for Parent Classes 39
8.14 ASCQM Release File Resource after Use in Operation 40
8.15 ASCQM Implement Virtual Destructor for Classes with Virtual Methods 41
8.16 ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls 42
8.17 ASCQM Ban Hard-Coded Literals used to Connect to Resource 42
8.18 ASCQM Ban Unintended Paths 43
8.19 ASCQM Ban While TRUE Loop Without Path To Break 44
8.20 ASCQM Ban Unmodified Loop Variable Within Loop 45
8.21 ASCQM Release File Resource after Use in Class 45
8.22 ASCQM Catch Exceptions 46
8.23 ASCQM Ban Empty Exception Block 47
8.24 ASCQM Ban Incompatible Lock Acquisition Sequences 48
8.25 ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues 48
8.26 ASCQM Ban Use of Prohibited Low-Level Resource Management Functionality 49
8.27 ASCQM Ban Excessive Size of Index on Columns of Large Tables 51
8.28 ASCQM Ban Excessive Number of Index on Columns of Large Tables 51
8.29 ASCQM Ban Excessive Complexity of Data Resource Access 52
8.30 ASCQM Ban Expensive Operations in Loops 53
8.31 ASCQM Limit Number of Aggregated Non-Primitive Data Types 55
8.32 ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure 55
8.33 ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code 56
8.34 ASCQM Ban Incremental Creation of Immutable Data 57
8.35 ASCQM Ban Unboxing in Loops 58
8.36 ASCQM Ban Autoboxing in Loops 59
8.37 ASCQM Implement Index Required by Query on Large Tables 61
8.38 ASCQM Release Memory after Use with Correct Reference 62

9 Calculation of ASCRSM and Functional Density Measures 65

© ISO/IEC 2025 – All rights reserved

9.1 Calculation of the Base Measures (Normative) 65
9.2 Functional Density of Weaknesses (Informative) 65
10 Alternative Weighted Measures and Uses (Informative) 66
11 References (Informative) 67
Annex A: Consortium for IT Software Quality (CISQ) 68
Annex B: Common Weakness Enumeration (CWE) (Informative) 69
Annex C: Relationship of the Automated Source Code Resource Sustainability Measure to ISO/IEC 25000 Series Standards (SQuaRE) (Informative) 70

© 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

1 Scope

1.1 Purpose

This is the third standard in the multipart Automated Source Code Quality Measures (ASCQM) standard (ISO/IEC 5055-1:2021). It covers common weaknesses (CWEs) that affect the use of energy and other resources. Specifying this measure is important as a source of evidence for complying with emerging regulations and corporate policies regarding reductions in resource usage. This measure is calculated from detecting and counting 33 violations of good architectural and coding practices (weaknesses) in the source code that could result in excessive or unnecessary processing or failures that cause hardware reboots.

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. They also provide evidence of compliance with regulations governing various aspects of software system performance.

Currently there are no standards for most of the software structural quality measures. ISO/IEC 25023 purports to address these measures but only provides measures of external behavior and does not define measures that can be developed from source code during development. This specification addresses one aspect of this problem by providing a specification for measuring attributes of the software that affect the efficient use of resources, often referred to as ‘Green IT’.

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 Resource Sustainability Measure is a correlated measure rather than an absolute measure of excessive resource usage. That is, since it does not measure all possible resource usage weaknesses, it does not provide an absolute measure of resource inefficiency. However, since it includes counts of what industry experts have determined to be the most severe weaknesses, it provides a strong indicator of the resource inefficiency of a software system. In most instances it will be highly correlated with the probability of inefficient resource usage.

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 presentation of automated techniques for detecting each weakness. 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 detectable by 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 each of the four Automated Source Code Quality Measures 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 triggered for each weakness, then summing these numbers for all weaknesses included in the Automated Source Code Resource Sustainability Measure. Each instantiation of a weakness

© ISO/IEC 2025 – All rights reserved

triggers only one of a weakness's detection patterns if multiple detection patterns are relevant to a weakness. Clauses 9 and 10 will present several methods for normalizing the results of evaluating this measure.

© ISO/IEC 2025 – All rights reserved

2 Conformance

Implementations of this specification shall demonstrate the following attributes in order 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***—Implementations of this specification shall state the assumptions and 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, 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.

Common Weakness Enumeration—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)

Detection Pattern—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)

Performance Efficiency—capability of a product to use an appropriate amount of resources under stated conditions (ISO/IEC 25010)

Quality Measure Element—measure defined in terms of a software quality attribute and the measurement method for quantifying it, including optionally the transformation by a mathematical function (ISO/IEC 25010)

Reliability—capability a product to perform specified functions under specified conditions for a specified period of time (ISO/IEC 25010)

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

Software Product Quality Model—model that categorizes software product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability) (ISO/IEC 25010)

Software Quality—degree to which a software product satisfies stated and implied needs when used under specified conditions (ISO/IEC 25010)

Software Quality Attribute—inherent property or characteristic of software that can be distinguished quantitatively or qualitatively by human or automated means (derived from ISO/IEC 25010)

Software Quality Characteristic—set of software quality attributes that affect a specific category of software quality outcomes (derived from ISO/IEC 25010)

Software Quality Characteristic Measure—software quality measure derived from measuring the attributes related to a specific software quality characteristic (ISO/IEC 25020)

Software Quality Measure—a measure that is defined as a measurement function of two or more values of software quality measure elements (ISO/IEC 25010)

Software Quality Measure Element—measure defined in terms of a software quality attribute and the measurement method for quantifying it, including optionally the transformation by a mathematical function (ISO/IEC 25010)

Software Quality Measurement—set of operations having the object of determining a value of a software quality measure (ISO/IEC 25010)

Software Quality Model—defined set of software characteristics, and of relationships between them, which provides a framework for specifying software quality requirements and evaluating the quality of a software product (derived from ISO/IEC 25010)

© ISO/IEC 2025 – All rights reserved

Software Quality Rule—architectural or coding practice or convention that represents good software engineering practice and avoids problems in software development, maintenance, or operations

Software Quality Sub-characteristic—sub-category of a software quality characteristic to which software quality attributes and their software quality measure elements are conceptually related (derived from ISO/IEC 25010)

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

Structural Quality—degree to which a set of static attributes of a software product satisfy stated and implied needs for the software product to be used under specified conditions (derived from ISO/IEC 25010)

Weakness—pattern or structure in the code (Detection Pattern in ASCRSM) 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

ASCPEM — Automated Source Code Performance Efficiency Measure

ASCQM — Automated Source Code Quality Measure

ASCRM — Automated Source Code Reliability Measure

ASCRSM — Automated Source Code Resource Sustainability 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