DRAFT INTERNATIONAL STANDARD

ISO/IEC DIS 5055

ISO/IEC JTC **1**	Secretariat: **ANSI**
Voting begins on: **2020-06-04**	Voting terminates on: **2020-08-27**

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

ICS: 35.080

This document is circulated as received from the committee secretariat.

FAST TRACK PROCEDURE

Reference number
ISO/IEC DIS 5055:2020(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Table of Contents

Table of Figures

Table of Tables

Preface

About the OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Meta-model); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at *https://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Formal Specifications are available from this URL: *https://www.omg.org/spec*

All of OMG"s formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
109 Highland Avenue
Suite 300
Needham, MA 02494
USA

Tel: +1-781-444-0404
Fax: +1-781-444-0320

Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO/IEC standards. Please consult: http://www.iso.org

Issues

The reader is encouraged to report and technical or editing issues/problems with this specification to the OMG main web page https://www.omg.org, under Documents, Report a Bug/Issue.

1 Scope

1.1 Purpose

This specification updates, expands, and combines four previous adopted specifications of the OMG:

- Automated Source Code Maintainability Measure (ASCMM) https://www.omg.org/spec/ASCMM/1.0/
- Automated Source Code Performance Efficiency Measure (ASCPEM) https://www.omg.org/spec/ASCPEM/1.0/
- Automated Source Code Reliability Measure (ASCRM) https://www.omg.org/spec/ASCRM/1.0/
- Automated Source Code Security Measure (ASCSM) https://www.omg.org/spec/ASCSM/1.0/

The measures in these standards 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 do not provide 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 only provides measures of external behavior and does not 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 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 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 in order 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
- MOF/XMI Mapping, version 2.5.1 (XMI), https://www.omg.org/spec/XMI/2.5.1/
- ISO/IEC 25010 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models
- ISO/IEC 25020:2007 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Measurement reference model and guide
- 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/stat**e** exchange **–** Common weakness enumeration

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)

Cyclomatic Complexity—A measure of control flow complexity developed by Thomas McCabe based on a graph-theoretic analysis that reduces the control flow of a computer program to a set of edges, vertices, and their attributes that can be quantified. (McCabe, 1976)

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

Internal Software Quality—the 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. This will be referred to as software structural quality, or simply structural quality in this specification. (ISO/IEC 25010)

Maintainability—capability of a product to be modified by the intended maintainers with effectiveness and efficiency (ISO/IEC 25010)

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—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 (ISO/IEC 25010)

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

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 Product Quality Model—a model that categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability). Each characteristic is composed of a set of related sub-characteristics. (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—an 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—a set of software quality attributes that affect a specific category of software quality outcomes. (similar to but more specific than ISO/IEC 25010)

Software Quality Characteristic Measure—a 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—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. (ISO/IEC 25010)

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

Software Quality Model—a 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)

Software Quality 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.

Software Quality Sub-characteristic—a 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—a component of software code that can be uniquely identified and counted such as a token, decision, variable, etc.

Structural Quality—the 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—a component of software quality. This concept is referred to as internal software quality in ISO/IEC 25010.

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)

5 Symbols (and Abbreviated Terms)

AFP — Automated Function Points

ASCMM — Automated Source Code Maintainability Measure

ASCPEM — Automated Source Code Performance Efficiency Measure

ASCQM — Automated Source Code Quality Measure

ASCRM — Automated Source Code Reliability Measure

ASCSM — Automated Source Code Security Measure

CWE — Common Weakness Enumeration

CISQ — Consortium for IT Software Quality

KDM — Knowledge Discovery Metamodel

SPMS — Structured Pattern Metamodel Standard

SMM — Structured Metrics Metamodel

