Date: September 2023

Automated Technical Debt Measure, Version 2

V1.0 – FTF2 Beta 2

OMG Document Number: ptc/23-09-04

Normative reference: http://www.omg.org/spec/ATDM/

Machine readable file(s):

Normative: http://www.omg.org/spec/ATDM/admtf-2023-02-02.xmi

This OMG document replaces formal/2018-09-01. Comments on the content of this document are welcome and should be directed to issues@omg.org by July 31, 2023.

You may view the pending issues for this specification from the OMG revision issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in October 2023. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page https://www.omg.org, under Documents, Report a Bug/Issue.

IPR Mode: Non-Assertive Covenant

Table of Contents

Preface

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 Metamodel); 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 Specifications are available from the OMG website at:
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
9C Medway Road, PMB 274
Milford, MA 01757
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*
Certain OMG specifications are also available as ISO standards. Please consult https://www.iso.org

1 Scope

1.1 Purpose

The purpose of this revised specification is to update *formal/18-09-01 Automated Technical Debt Measure*. This revision is based on updates to the Weaknesses included in the four quality measures defined in *formal/2020-01-02 Automated Source Code Quality Measures*, that was subsequently approved and published as *ISO/IEC 1055:2021 Automated Source Code Quality Measures*. The technical objective of this specification and its predecessor are to establish a standard for automating a measure of Technical Debt that can be computed by source code analysis technologies which have implemented *formal/2020-01-02 Automated Source Code Quality Measures*. Within the context of these four quality measures, Technical Debt is calculated as an estimate of the effort to fix Weaknesses that represent violations of good architectural and coding practices that must be corrected because of their risk and/or cost to the owner or user of the software system. Currently, several static analysis vendors calculate a proprietary measure of Technical Debt. Using OMG and ISO standards to provide the content for a measure of Technical Debt allows it to be based on published standards rather than proprietary calculations.

1.2 The Technical Debt Metaphor

The Technical Debt metaphor was introduced by Ward Cunningham to describe how sub-optimal design decisions, often made to meet release schedules, accumulated a debt that had to be repaid through corrective maintenance during future releases. CISQ participated in a 2016 workshop in Dagstuhl, Germany along with 40 members of the Technical Debt research community to create a framework for defining the metaphor and guiding research (Curtis, 2016). Two conclusions were reached at the end of the week.

1) There is no universally agreed definition of Technical Debt.
2) Industry and the research community have different goals in using a Technical Debt measure.

Regarding the second point, many in the research community restrict the domain of Technical Debt to consciously sub-optimal design decisions that primarily affect maintainability issues such as changeability and scalability. Consistent with Cunningham's original formulation of the concept, they do not consider missing features, functional defects, or most structural flaws related to reliability, security, or performance efficiency to be part of the Technical Debt domain. The participants in the Dagstuhl workshop were unable to construct a crisp definition delimiting the domain of Weaknesses to be included in Technical Debt.

In contrast, industry wants a measure that predicts the future costs of corrective maintenance and other software quality-related outcomes. Since the Consortium for IT Software Quality (CISQ) is an industry consortium, it has developed a specification for Technical Debt that is designed to predict corrective maintenance costs for guiding IT decisions and resource allocations. This measure of Technical Debt builds on *formal/2020-01-02 Automated Source Code Quality Measures* that CISQ developed for measuring the structural quality of software in the areas of Reliability, Security, Performance Efficiency, and Maintainability.

Choosing 'debt' as a metaphor engages a set of financial concepts that help executives think about software quality in business terms. The components that comprise Technical Debt provide a

foundation for the economics of software quality. The metaphor can be partitioned into the following elements which are displayed in Figure 1.

- ***Technical Debt***—Future costs attributable to known structural Weaknesses in production code that must be fixed. Technical Debt includes both the debt's principal and interest. A Weakness in production code is only included in Technical Debt calculations if those responsible for the application believe it is a 'must-fix' problem, therefore incurring corrective maintenance costs in a future release. Technical Debt is a primary component of the cost of application ownership.
- ***Principal***—The cost of remediating must-fix problems in production code. At a minimum, the principal is calculated from the number of hours required to correct these problems, multiplied by the fully burdened hourly cost of those involved in designing, implementing, and unit testing these fixes.
- ***Interest***—Continuing costs, primarily in IT, attributable to must-fix problems so long as they remain in production code. These ongoing costs can result from the excessive effort to modify unnecessarily complex code, greater resource usage by inefficient code, etc.
- ***Business Risk***—Potential costs to the business if must-fix problems in production code cause damaging operational events such as outages, data corruption, performance degradation, and security breaches.
- ***Liability***—Costs to the business resulting from operational problems caused by flaws in production code. These flaws include both must-fix problems included in the calculation of Technical Debt as well as problems not listed as must-fix because their risk was underestimated.
- ***Opportunity Cost***—Benefits such as revenue from new features that could have been achieved had resources been committed to developing new capability rather than being assigned to retire Technical Debt. Opportunity costs represent the tradeoff that application managers and executives must weigh when deciding how much effort to devote to retiring Technical Debt.

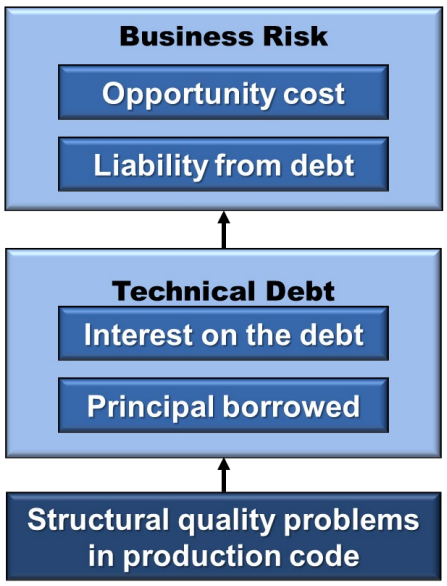

Figure 1. The Technical Debt metaphor

The cost to fix structural quality problems constitutes the principal of the debt, while the inefficiencies they cause such as greater maintenance effort or excessive computing resources represent interest costs on the debt. The structural problems underlying Technical Debt also create business risks such as outages and security breaches, and the negative events they can cause result in liabilities such as lost revenue from online sales or costly clean-up from a security breach. The effort spent to correct Technical Debt rather than develop new business functionality represents opportunity costs related to lost benefits that might otherwise have been achieved.

Adoption of the Technical Debt metaphor has provided a means of communicating between IT executives and their technical staffs about the costs of quality problems. Commercial IT executives have embraced the concept of Technical Debt for its value in predicting such factors as the costs of future corrective maintenance, the risks of operational problems, and the difficulty of enhancing or scaling applications.

1.3 Measuring Technical Debt

This specification is narrowly focused on defining a measure of the principal of Technical Debt that can be computed from the source code Weaknesses included *formal/202020-01-02 Automated Source Code Quality Measures*. Other components of the Technical Debt metaphor may become the focus of future OMG specifications. There are five steps in calculating Technical Debt that form the normative component of this specification.

1. Detect Occurrences of Detection Patterns that constitute Weaknesses in the Automated Source Code Quality Measures specifications (covering Reliability, Security, Performance Efficiency, and Maintainability quality characteristics); that is, detect the 135 unique violations of good architectural and coding practices from which these measures are calculated.
2. Assign an estimate of the amount of time to correct each of the 135 Detection Patterns based on a survey of software professionals; the estimate is a constant applied to each Occurrence of a source code Detection Pattern designed to detect Weaknesses.
3. Collect information about the structural context within which each Occurrence of a Detection Pattern is embedded.
4. Compute an Adjustment Factor based on the structural context surrounding each Occurrence of a Detection Pattern to adjust the estimate of its Remediation Effort time.
5. Sum the total amount of Remediation Effort time across all the Occurrences of each Detection Pattern for all 135 Detection Patterns to aggregate them into an estimate of total Technical Debt.
6. A separate Technical Debt Measure can be computed for each of the four quality characteristics defined in the Automated Source Code Quality Measures (ASCQM) standard (Reliability, Security, Performance Efficiency, and Maintainability) by aggregating the estimated Remediation Effort times separately for the Detection Patterns underlying Weaknesses included in each quality characteristic.

This normative specification does not include variations in labor costs, skill levels, or currencies (dollars, euros, rupees, etc.) since these are adjustments that must be made based on local conditions. The specification will include a set of non-normative usage scenarios showing how contextual information from step 3 can be used to manage Technical Debt measures as well as customize the Technical Debt measure to local conditions within an organization. These factors include issues related to system testing and other processes that can vary across organizations.

1.4 Technical Debt as an Estimate

Technical Debt measures are most frequently used to estimate future corrective maintenance costs as input to decisions such as budgeting maintenance, allocating developer effort, or replacing an application. Corrective maintenance includes all the activities involved in analyzing a Weakness, designing, implementing, and unit testing a Remediation. Costs for subsequent activities such as integration testing, system testing, and deployment are typically aggregated across the Remediations of numerous Weaknesses. Consequently, these costs are difficult if not impossible to separate and allocate among Weakness.

The measure defined in this specification is a correlated rather than absolute measure of Technical Debt. That is, it is a predictor of the amount of corrective maintenance effort needed for an application. Each organization must develop its own equation linking Technical Debt with its costs and other outcomes. There are three primary issues that affect the usefulness of this measure.

First, the Weaknesses incorporated in the four Automated Source Code Quality Measures (Reliability, Security, Performance Efficiency, and Maintainability) were selected because they were considered Weaknesses of sufficient severity that they should be corrected because of their risk to costs and operational performance. However, an organization may choose to correct only some of these Weaknesses, thereby not incurring the debt associated with other Weaknesses. In this case the Technical Debt measure will over-estimate corrective maintenance costs. Conversely, an organization can choose to correct more Weaknesses than are included in the four quality measures, in which case Technical Debt underestimates corrective maintenance costs. In either case, an organization can compute a Contextual Technical Debt Measure (CTDM) that better estimates their specific Remediation Effort the Automated Technical Debt Measure (ATDM) provides a common benchmark for comparing the structural quality and Remediation Effort across industry, while Contextual Technical Det Measure (CTDM) provides an organizational or application specific modification that better evaluates Remediation Effort based on the unique attributes of local quality assurance strategies.

Second, there are no existing industry-wide repositories of effort data related to correcting violations of good architectural and coding practices. Consequently, the Remediation Effort times used in this specification are based on surveys of experienced developers. The survey requested developers to estimate their time-to-fix for the 138 Weaknesses included in the four quality measures. The estimated times were to include Remediation Effort activities up to and including unit test. Most respondents were primarily developing in Java, .NET, or C# applications. Therefore, these Remediation Effort times are primarily relevant to business software rather than embedded software and languages such a C and C++. Default Remediation Effort times were developed as a constant for each Weakness from the modal tendency of the distribution of Remediation Effort times for that Weakness. The values at the Weakness level are then used at the underlying level, the level of the 135 Detection Patterns.

Variations in time estimates and sampling factors could impact the default Remediation Effort times drawn from these data. Consequently, the specification allows for these default times to be overridden with local estimates where appropriate. As more data become available, these default constants can be updated, if necessary, in a future revision of this specification. The Remediation Effort times for each violation are adjusted using the contextual information discussed in later clauses. Similarly, these Adjustment Factors can be updated in future revisions as data become available regarding their value in improving estimates of Remediation Effort times.

Third, Technical Debt measures are developed from static analysis of the Weaknesses in the structural quality of an application. It does not measure functional defects which must be corrected. Therefore, this measure does not assess all factors contributing to corrective maintenance costs.

However, since practices related to detecting the non-functional, structural Weaknesses in software have lagged those focused on functional defects, future maintenance effort is most often focused on structural Weaknesses. Consequently, Technical Debt provides an estimate of these costs that can be adjusted to account for local experience in remediating functional defects that escape testing and must be fixed in future releases.

In view of these considerations, Technical Debt provides an estimate based on OMG standards that can used to predict future risk and cost outcomes for a software-intensive system. It can be used as a benchmark for comparing applications that can be adjusted to local quality assurance practices and strategies.

2 Conformance

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

- ***Automated—***The calculation of this measure shall be fully automated. A conformant technology shall be able to consume and process machine readable outputs reporting Weaknesses detected from analysis of at least one of the four Automated Source Code Quality Measures and elements from analysis of the Automated Enhancement Points measure. Analyses to develop these inputs shall be provided with the software source code, the artifacts and information needed to configure the software for operation, and any available description of the architectural layers in the software.
- ***Complete***––A conformant technology shall be able to calculate a Technical Debt measure as specified in this document. Consequently, the technology used to compute this measure shall be able to receive and process outputs produced by technologies that comply with calculating the measures in *formal/2020-01-02 Automated Source Code Quality Measures* and *formal/17-04-03 Automated Enhancement Points*.
- ***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 tools that supplied inputs to this measure, as well as the source code, non-source code artifacts, and other information used by these other tools to prepare the source code for analysis.
- ***Verifiable—***A conformant implementation shall state the assumptions and heuristics it uses in computing this measure in sufficient detail that the calculations can be independently verified by third parties. Sub-clause 80 describes the measures and information required in the generated output. In addition, all inputs used are required to be clearly described and itemized so that they can be audited by a third party.

3 References

3.1 Normative References

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

List of normative references.

- Knowledge Discovery Meta-model, version 1.3 (KDM), formal/2011-08-04
- Structured Metrics Meta-model, version 1.1 (SMM), formal/2015-10-03
- Meta Object Facility, version 2.5 (MOF), formal/2015-06-05
- XML Metadata Interchange, version 2.5.1 (XMI), formal/2015-06-07
- Object Constraint Language, version 2.4 (OCL), formal/2014-02-03
- Automated Source Code Quality Measures, formal/2020-01-02
- ISO/IEC 5055:2021 Information technology — Software measurement — Software quality measurement — Automated source code quality measures
- Automated Enhancement Points, version 1.0 (AEP), formal/2017-04-03
- Structured Patterns Metamodel Specification 1.0 (SPMS), formal/2015-10-01
- ISO/IEC 25010:2011 Systems and software engineering - System and software product Quality Requirements and Evaluation (SQuaRE) - System and software quality models

3.2 Non-normative References

List of non-normative references.

- Paris Avgeriou, Philippe Kruchten, Robert L. Nord, Ipek Ozkaya, Carolyn Seaman (2016). Reducing friction in software development. *IEEE Software*, 33 (1), 66-73.
- Consortium for IT Software Quality (2017). *CISQ Compliance Assessment*. Needham, MA: Object Management Group.
- Consortium for IT Software Quality (2022). *CISQ Time-to-Fix Survey*. Needham, MA: Object Management Group.
- Ward Cunningham (1992). The WyCash Portfolio Management System, *OOPSLA '92 Experience Report.*
- Curtis, B. (2016). Measuring and communicating the technical debt metaphor in industry. *Managing Technical Debt in Software Engineering*. Dagstuhl, Germany: Dagstuhl Publishing, 121-122.
- B. Curtis, J. Sappidi, & A. Szynkarski, (2012). Estimating the principal of an application's technical debt. *IEEE Software, 29* (6), 34-42.
- P. Kruchten, R. L. Nord, I. Ozkaya (2012). Technical Debt: From Metaphor to Theory and Practice. *IEEE Software*, 29 (6), 30-33.
- Software Engineering Body of Knowledge, V3.0 (SWEBOK). http://www.computer.org/web/swebok/v3

4 Terms and Definitions

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

Adjusted Remediation Effort

The number of minutes needed to correct a specific source code pattern that has been adjusted by contextual measures.

Adjusted Technical Debt

A measure of Technical Debt that:

- only measures Technical Debt Items that are a selected subset of the Detection Patterns included in Technical Debt, and/or
- uses a Remediation Effort configuration different from the one described in this specification, and/or
- incorporates an Adjustment Factor as presented in the normative Sub-clause 7.11, and/or
- incorporates contextual factors such as the ones presented in the informative Clause 9.

Application Model

A representation of the Application composed of the computational objects in the source code and their relationships, some of which can contain processing rules and logic.

[SOURCE: *formal/2011-08-04 Knowledge Discovery Meta-model, version 1.3*]

Automated Maintainability Remediation Effort

The sum of Remediation Efforts of all detected Technical Debt Items associated with Weaknesses in the Automated Source Code Maintainability Measure specification *(formal/2020-01-02)*.

Automated Performance Efficiency Remediation Effort

The sum of Remediation Efforts of all detected Technical Debt Items associated with Weaknesses included in the Automated Source Code Performance Efficiency Measure specification *(formal/2020-01-02)*.

Automated Reliability Remediation Effort

The sum of Remediation Efforts of all detected Technical Debt Items associated with Weaknesses included in the Automated Source Code Reliability Measure specification *(formal/2020-01-02)*.

Automated Security Remediation Effort

The sum of Remediation Efforts of all detected Technical Debt Items associated with Weaknesses included in the Automated Source Code Security Measure specification *(formal/2020-01-02)*.

Automated Source Code Quality Measures

Four measures derived from static analysis of software source code that quantify the number of severe Weaknesses affecting the Reliability, Security, Performance Efficiency, and Maintainability of a software system.

[SOURCE: *formal/2020-01-02/Automated Source Code Quality Measures* and *ISO/IEC 5055:2021 Automated Source Code Quality Measures*]

Note 1 to entry: The definition of each measure conforms to the definition of its related quality characteristic in *ISO/IEC 25010:2011 System and Software Quality Model*.

Automated Technical Debt

The sum of Remediation Efforts of all detected Technical Debt Items associated with Weaknesses enumerated in the Automated Source Code Quality Measures specifications *(formal/2020-01-02)*.

Code Element

A collection of programming language instructions such as a class, module, component, or subroutine treated as a single grouping of code for functions such as compiling or unit testing.

Complexity [or Effort Complexity]

Contextual Information regarding the code elements implementing the Occurrence of a Detection Pattern measured according to the Effort Complexity definition from the Automated Enhancement Points specification.

[SOURCE: *formal/2017-04-03 Automated Enhancement Points, version 1.0*]

Concentration

Contextual Information which measures the number of Occurrences of Detection Patterns within any Code Element in the software.

Contextual information

Attributes of the software context affecting a specific Occurrence of a Detection Pattern that can cause variation in the time required to correct the Occurrence.

Contextual measures

Quantifications of the Contextual Information used in adjusting calculations of Remediation Effort in the Automated Technical Debt Measure.

Contributing Weakness

A Weakness that represents a conceptually distinct Instantiation of a Parent Weakness based on differences in the Detection Patterns underlying each instantiation.

Corrective Maintenance

All the activities involved in analyzing a Weakness, designing and implementing a Remediation, and unit testing it.

Detection Pattern

A collection of parsed program elements and their relations that constitute a Weakness in the software and can be identified through automated matching of a Detection Pattern with structures in the source code.

Note 1 to entry: Detection Patterns for the 138 Weaknesses incorporated in this specification are enumerated in *ISO/IEC 5055:2021.*

Detection Pattern Occurrence

A single instance of a Detection Pattern that has been implemented in the software.

Detection Pattern Role

Pattern roles describe the set of code elements involved in relationships that create a Detection Pattern.

[SOURCE: *formal/2015-10-01 Structured Patterns Metamodel Specification 1.0*]

Note 1 to entry: A Role is a required element in a Structured Pattern Definition

Technological Diversity

Contextual Information which measures the number of distinct programming languages in which the source code elements included in a single Occurrence of a Detection Pattern are written.

Evolution

The Evolution of the Occurrence of a Weakness and of code elements implementing the Occurrence is contextual information indicating if the Occurrence of a Weakness or the code elements implementing it have been added, updated, or deleted between measured revisions of the software.

Exposure

Contextual Information measuring the level of connectedness of an Occurrence of a Weakness with the rest of the software, both directly and indirectly through call paths.

Occurrence

A single instance of a Detection Pattern for a Weakness occurring in the source code.

Occurrence Gap Size

The difference between the actual measured values and specified threshold values for Detection Patterns incorporating a role that defines threshold values that are not to be exceeded.

Parent Weakness

A Weakness that has several conceptually distinct instantiations based on differences in the Detection Patterns underlying each instantiation.

Remediation Effort

The time required to correct an Occurrence, or set of Occurrences, of a Detection Pattern in the software.

Software Cost

Money spent for developing, correcting, or enhancing the software. In this specification it is the money spent on corrective maintenance.

Software Value

The business benefit derived by the owners or users of the software.

Software Quality

The degree to which the software meets customer or user needs or expectations, and is free of defects that could cause the software to fail to meet these needs or expectations in the future.

[SOURCE: *ISO/IEC 25010:2011 System and Software Quality Model*]

Technical Debt Item

An atomic constitutive element of Technical Debt, that is, a single Occurrence of a Detection Pattern in the source code associated with one of the 138 Weaknesses enumerated in *ISO/IEC 5055:2021*.

Unadjusted Remediation Effort

The number of minutes needed to correct a specific source code pattern before being adjusted by Contextual measures.

Note 1 to entry: Default Unadjusted Remediation Efforts have been assigned to each of the 138 Detection Patterns defined in *ISO/IEC 5055:2021.*

Note 2 to entry: Default Unadjusted Remediation Efforts values can be changed to better fit the local context and conditions prior to calculating a Technical Debt measure.

Weakness

A non-conformity to good architectural and coding practices.

Note 1 to entry: In this specification ‘Weaknesses’ refer to the 138 Weaknesses enumerated in *ISO/IEC 5055:2021.*

.

5 Symbols

List of symbols/abbreviations.

AEP	Automated Enhancement Points
MREM	Automated Maintainability Remediation Effort Measure
PEREM	Automated Performance Efficiency Remediation Effort Measure
RREM	Automated Reliability Remediation Effort Measure
SREM	Automated Security Remediation Effort Measure
ASCMM	Automated Source Code Maintainability Measure
ASCPEM	Automated Source Code Performance Efficiency Measure
ASCQM	Automated Source Code Quality Measures
ASCRM	Automated Source Code Reliability Measure
ASCSM	Automated Source Code Security Measure
ATDM	Automated Technical Debt Measure
CISQ	Consortium for IT Software Quality
CTDM	Contextual Technical Debt Measure
KDM	Knowledge Discovery Metamodel
OCL	Object Constraint Language
SMM	Structured Metrics Metamodel
SPMS	Structured Patterns Metamodel Specification
TD	Technical Debt

6 Foundational Information (Informative)

6.1 Automated Source Code Quality Measures

The Automated Technical Debt Measure (ATDM) is calculated from Occurrences of the 138 Weaknesses included in the four Quality Characteristic measures contained in OMG's Automated Source Code Quality Measures (ASCQM), which has been approved and published as ISO/IEC 5055:2021. Detecting and counting these Weaknesses is the starting point for calculating ATDM. The Automated Source Code Quality Measures consist of the following measures contained in ASCQM and ISO/IEC 5055:2021.

- ***Automated Source Code Reliability Measure (ASCRM)*** — violations of good architectural and coding practice that can cause outages, delayed recovery, data corruption, and unpredictable operational behavior.
- ***Automated Source Code Security Measure (ASCSM)*** — violations of good architectural and coding practice in an application that allow unauthorized intrusion into the application's source code, data store, operations, or connections.
- ***Automated Source Code Performance Efficiency Measure (ASCPEM)*** — violations of good architectural and coding practice that can result in slow response, degraded performance, or excessive use of computational resources.
- ***Automated Source Code Maintainability Measure (ASCMM)*** — violations of good architectural and coding practice that make an application's source code difficult to understand or modify.

The following sub-clauses provide additional background information about the scope and content of Automated Source Code Quality Measure specifications regarding:

- the nature of development artifacts involved,
- the identification of Occurrences of Detection Patterns from the *ASCMM*, *ASCRM*, *ASCPEM*, and *ASCSM* specifications, including the modeling of the effort associated with remediating an actual Technical Debt Item, or
- the Contextual Information associated with each Detection Pattern Occurrence, that is, additional information associated with the Occurrence to aid in prioritizing its Remediation and other decisions or estimates.

6.1.1 Development artifacts

Development artifacts associated with a Technical Debt item can be found in various locations:

- **Source Code**, including implemented Software Structure and Architecture
- **Build Scripts**
- **Test Scripts**
- **Documentatio**n
- **Technology**
- **Design**, including Architecture Decisions

6.1.1.1 Source Code

Source Code Development artifacts include all the elements and inter-element relationships that exist in the source code and the Application Model produced from it. The Application Model allows

automated tools to analyze the software structure and architecture as implemented in the source code, rather than how the structure and architecture were designed or documented. Source Code Development artifacts are represented by the following elements from the Knowledge Discovery Metamodel (*KDM*):

- **Source package—**representing physical artifacts,
- **Code package—**representing low-level building blocks of the software,
- **Action package—**representing low-level relationships and statements,
- **Platform package—**representing run-time resources,
- **UI package—**representing user-interface aspects of the software,
- **Event package—**representing event-driven aspects of the software,
- **Data package—**representing persistent data aspects of the software,
- **Structure package—**representing architectural components of the software.

6.1.1.2 Build Scripts

Build Scripts Development artifacts include all the elements produced by development teams to build the software. Build Scripts Development artifacts are represented by the following elements from the Knowledge Discovery Meta-model (*KDM*):

- **Build package—**representing artifacts related to the build process,
- **Source and Code packages—**used as build resources.

6.1.1.3 Test Scripts

Test Scripts Development artifacts include all the elements produced by development teams to verify the correct functioning of the software. Test Scripts Development artifacts are represented by the same *KDM* packages as Source Code Development artifacts, and only differ in nature by the intent behind their production.

6.1.1.4 Documentation

Documentation Development artifacts include all the elements produced by development teams to help understand how the software was developed. They do not include documentation artifacts that are found in the source code, and that are already covered by Source Code Development artifacts.

6.1.1.5 Technology

Technology Development artifacts are the programming languages used in developing the software, as well as third party supplied components that are required to develop and execute the software. In other words, they include all elements used in the software which are not under the control of the development organization, but can negatively impact the software or its development process. For example, the Technical Debt created by the discontinuation of the technologies used in developing the software.

6.1.1.6 Design

Design Development artifacts are all the decisions, including architectural decisions made and documented prior to developing the code. Design Development artifacts do not include the software design and architectural elements that are determined by analyzing the source code.

6.1.2 Source Code Weaknesses

The Automated Source Code Quality Measure (ASCQM) specification defines lists of source code Weaknesses for each of the four quality measures that are considered severe enough violations of good architectural and coding practice that they should be corrected in a near-term release. These

Weaknesses are specified in formats drawn from the Structured Patterns Metamodel Specification (SPMS) using element descriptions from the Knowledge Discovery Metamodel (KDM). Each Weakness constitutes a Technical Debt measurement Item and is listed by its Common Weakness Enumeration identifier (CWE ###). Some Weaknesses may be associated with more than one quality measure. There are 138 unique Weaknesses included in ASCQM.

Tables 1-4 provide the list of Weaknesses for each of the quality characteristic measures in ASCQM. In these tables Weaknesses are identified as either a parent or contributing Weakness. Contributing Weaknesses are unique instantiations of a parent Weakness. CWE numbers for parent Weakness are presented in dark blue cells. CWE numbers for contributing Weaknesses are presented in light blue cells immediately below their parent Weakness.

6.2 Automated Source Code Security Measure (ASCSM) Weaknesses

ASCSM contains 36 parent Weaknesses and 37 contributing Weaknesses which are presented in Table 1. CWE #s for Contributing Weaknesses are presented in light blue immediately following their Parent Weakness whose CWE # is presented in dark blue.

Table 1: List of ASCSM Weaknesses

CWE #	Descriptor
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
CWE-23	Relative Path Traversal
CWE-36	Absolute Path Traversal
CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CWE-88	Argument Injection or Modification
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
CWE-564	SQL Injection: Hibernate
CWE-90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')
CWE-91	XML Injection (aka Blind XPath Injection)
CWE-99	Improper Control of Resource Identifiers ('Resource injection')
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CWE-123	Write-what-where condition
CWE-125	Out-of-bounds Read
CWE-130	Improper Handling of Length Parameter Inconsistency

CWE-786	Access of Memory Location Before Start of Buffer
CWE-787	Out-of-bounds Write
CWE-788	Access of Memory Location After End of Buffer
CWE-805	Buffer Access with Incorrect Length Value
CWE-822	Untrusted Pointer Dereference
CWE-823	Use of Out-of-range Pointer Offset
CWE-824	Access of Uninitialized Pointer
CWE-825	Expired Pointer Dereference
CWE-129	Improper Validation of Array Index
CWE-134	Use of Externally Controlled Format String
CWE-252	Unchecked Return Value
CWE-404	Improper Resource Shutdown or Release
CWE-401	Improper Release of Memory Before Removing Last Reference ('Memory Leak')
CWE-772	Missing Release of Resource after Effective Lifetime
CWE-775	Missing Release of File Descriptor or Handle after Effective Lifetime
CWE-424	Improper Protection of Alternate Path
CWE-434	Unrestricted Upload of File with Dangerous Type
CWE-477	Use of Obsolete Function
CWE-480	Use of Incorrect Operator
CWE-502	Deserialization of Untrusted Data
CWE-570	Expression is Always False
CWE-571	Expression Is Always True
CWE-606	Unchecked Input for Loop Condition
CWE-611	Improper Restriction of XML External Entity Reference ('XXE')
CWE-643	Improper Neutralization of Data within XPath Expressions ('XPath Injection')
CWE-652	Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')
CWE-665	Improper Initialization
CWE-456	Missing Initialization of a Variable
CWE-457	Use of uninitialized variable
CWE-662	Improper Synchronization
CWE-366	Race Condition within a Thread
CWE-543	Use of Singleton Pattern Without Synchronization in a Multithreaded Context
CWE-567	Unsynchronized Access to Shared Data in a Multithreaded Context
CWE-667	Improper Locking
CWE-820	Missing Synchronization
CWE-821	Incorrect Synchronization
CWE-672	Operation on a Resource after Expiration or Release
CWE-415	Double Free
CWE-416	Use After Free

CWE-681	Incorrect Conversion between Numeric Types
CWE-194	Unexpected Sign Extension
CWE-195	Signed to Unsigned Conversion Error
CWE-196	Unsigned to Signed Conversion Error
CWE-197	Numeric Truncation Error
CWE-682	Incorrect Calculation
CWE-131	Incorrect Calculation of Buffer Size
CWE-369	Divide By Zero
CWE-732	Incorrect Permission Assignment for Critical Resource
CWE-778	Insufficient Logging
CWE-783	Operator Precedence Logic Error
CWE-789	Uncontrolled Memory Allocation
CWE-798	Use of Hard-coded Credentials
CWE-259	Use of Hard-coded Password
CWE-321	Use of Hard-coded Cryptographic Key
CWE-835	Loop with Unreachable Exit Condition ('Infinite Loop')
CWE-917	Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')
CWE-1057	Data Access Operations Outside of Expected Data Manager Component

6.3 Automated Source Code Reliability Measure (ASCRM) Weaknesses

ASCRM contains 35 parent Weaknesses and 39 contributing Weaknesses as presented in Table 2. CWE #s for Contributing Weaknesses are presented in light blue immediately following their Parent Weakness whose CWE # is presented in dark blue.

Table 2: List of ASCRM Weaknesses

CWE #	Descriptor
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CWE-123	Write-what-where condition
CWE-125	Out-of-bounds read
CWE-130	Improper Handling of Length Parameter Inconsistency
CWE-786	Access of Memory Location Before Start of Buffer
CWE-787	Out-of-bounds Write
CWE-788	Access of Memory Location After End of Buffer
CWE-805	Buffer Access with Incorrect Length Value
CWE-822	Untrusted Pointer Dereference
CWE-823	Use of Out-of-range Pointer Offset
CWE-824	Access of Uninitialized Pointer
CWE-825	Expired Pointer Dereference

CWE-170	Improper Null Termination
CWE-252	Unchecked Return Value
CWE-390	Detection of Error Condition Without Action
CWE-394	Unexpected Status Code or Return Value
CWE-404	Improper Resource Shutdown or Release
CWE-401	Improper Release of Memory Before Removing Last Reference ('Memory Leak')
CWE-772	Missing Release of Resource after Effective Lifetime
CWE-775	Missing Release of File Descriptor or Handle after Effective Lifetime
CWE-424	Improper Protection of Alternate Path
CWE-459	Incomplete Clean-up
CWE-476	NULL Pointer Dereference
CWE-480	Use of Incorrect Operator
CWE-484	Omitted Break Statement in Switch
CWE-562	Return of Stack Variable Address
CWE-595	Comparison of Object References Instead of Object Contents
CWE-597	Use of Wrong Operator in String Comparison
CWE-1097	Persistent Storable Data Element without Associated Comparison Control Element
CWE-662	Improper Synchronization
CWE-366	Race Condition within a Thread
CWE-543	Use of Singleton Pattern Without Synchronization in a Multithreaded Context
CWE-567	Unsynchronized Access to Shared Data in a Multithreaded Context
CWE-667	Improper Locking
CWE-764	Multiple Locks of a Critical Resource
CWE-820	Missing Synchronization
CWE-821	Incorrect Synchronization
CWE-1058	Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element
CWE-1096	Singleton Class Instance Creation without Proper Locking or Synchronization
CWE-665	Improper Initialization
CWE-456	Missing Initialization of a Variable
CWE-457	Use of uninitialized variable
CWE-672	Operation on a Resource after Expiration or Release
CWE-415	Double Free
CWE-416	Use After Free
CWE-681	Incorrect Conversion between Numeric Types
CWE-194	Unexpected Sign Extension
CWE-195	Signed to Unsigned Conversion Error
CWE-196	Unsigned to Signed Conversion Error
CWE-197	Numeric Truncation Error

CWE-682	Incorrect Calculation
CWE-131	Incorrect Calculation of Buffer Size
CWE-369	Divide By Zero
CWE-703	Improper Check or Handling of Exceptional Conditions
CWE-248	Uncaught Exception
CWE-391	Unchecked Error Condition
CWE-392	Missing Report of Error Condition
CWE-704	Incorrect Type Conversion or Cast
CWE-758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior
CWE-833	Deadlock
CWE-835	Loop with Unreachable Exit Condition ('Infinite Loop')
CWE-908	Use of Uninitialized Resource
CWE-1045	Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor
CWE-1051	Initialization with Hard-Coded Network Resource Configuration Data
CWE-1066	Missing Serialization Control Element
CWE-1070	Serializable Data Element Containing non-Serializable Item Elements
CWE-1077	Floating Point Comparison with Incorrect Operator
CWE-1079	Parent Class without Virtual Destructor Method
CWE-1082	Class Instance Self Destruction Control Element
CWE-1083	Data Access from Outside Designated Data Manager Component
CWE-1087	Class with Virtual Method without a Virtual Destructor
CWE-1088	Synchronous Access of Remote Resource without Timeout
CWE-1098	Data Element containing Pointer Item without Proper Copy Control Element

6.4 Automated Source Code Performance Efficiency Measure (ASCPEM) Weaknesses

ASCPEM contains 16 parent Weaknesses and 3 contributing Weaknesses as presented in Table 3. CWE #s for Contributing Weaknesses are presented in light blue immediately following their Parent Weakness whose CWE # is presented in dark blue.

Table 3: List of ASCPEM Weaknesses

CWE #	Descriptor
CWE-404	Improper Resource Shutdown or Release
CWE-401	Improper Release of Memory Before Removing Last Reference ('Memory Leak')
CWE-772	Missing Release of Resource after Effective Lifetime
CWE-775	Missing Release of File Descriptor or Handle after Effective Lifetime
CWE-424	Improper Protection of Alternate Path
CWE-1042	Static Member Data Element outside of a Singleton Class Element

CWE-1043	Data Element Aggregating an Excessively Large Number of Non-Primitive Elements
CWE-1046	Creation of Immutable Text Using String Concatenation
CWE-1049	Excessive Data Query Operations in a Large Data Table
CWE-1050	Excessive Platform Resource Consumption within a Loop
CWE-1057	Data Access Operations Outside of Expected Data Manager Component
CWE-1060	Excessive Number of Inefficient Server-Side Data Accesses
CWE-1067	Excessive Execution of Sequential Searches of Data Resource
CWE-1072	Data Resource Access without Use of Connection Pooling
CWE-1073	Non-SQL Invokable Control Element with Excessive Number of Data Resource Accesses
CWE-1089	Large Data Table with Excessive Number of Indices
CWE-1091	Use of Object without Invoking Destructor Method
CWE-1094	Excessive Index Range Scan for a Data Resource

6.5 Automated Source Code Maintainability Measure (ASCMM) Weaknesses

ASCMM contains 29 parent Weaknesses and no contributing Weaknesses.

Table 4: List of ASCMM Weaknesses

CWE #	Descriptor
CWE-407	Algorithmic Complexity
CWE-478	Missing Default Case in Switch Statement
CWE-480	Use of Incorrect Operator
CWE-484	Omitted Break Statement in Switch
CWE-561	Dead code
CWE-570	Expression is Always False
CWE-571	Expression is Always True
CWE-783	Operator Precedence Logic Error
CWE-1041	Use of Redundant Code (Copy-Paste)
CWE-1045	Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor
CWE-1047	Modules with Circular Dependencies
CWE-1048	Invokable Control Element with Large Number of Outward Calls (Excessive Coupling or Fan-out)
CWE-1051	Initialization with Hard-Coded Network Resource Configuration Data
CWE-1052	Excessive Use of Hard-Coded Literals in Initialization
CWE-1054	Invocation of a Control Element at an Unnecessarily Deep Horizontal Layer (Layer-skipping Call)
CWE-1055	Multiple Inheritance from Concrete Classes
CWE-1062	Parent Class Element with References to Child Class
CWE-1064	Invokable Control Element with Signature Containing an Excessive Number of Parameters

CWE-1074	Class with Excessively Deep Inheritance
CWE-1075	Unconditional Control Flow Transfer outside of Switch Block
CWE-1079	Parent Class without Virtual Destructor Method
CWE-1080	Source Code File with Excessive Number of Lines of Code
CWE-1084	Invokable Control Element with Excessive File or Data Access Operations
CWE-1085	Invokable Control Element with Excessive Volume of Commented-out Code
CWE-1086	Class with Excessive Number of Child Classes
CWE-1087	Class with Virtual Method without a Virtual Destructor
CWE-1090	Method Containing Access of a Member Element from Another Class
CWE-1095	Loop Condition Value Update within the Loop
CWE-1121	Excessive McCabe Cyclomatic Complexity

6.6 Source Code Pattern Roles

Each Weakness definition contains a specification of Roles (**SPMS:Definitions::Roles**). According to the Structured Patterns Metamodel Specification (SPMS), "A pattern is informally defined as a set of relationships between a set of entities. Roles describe the set of entities within a pattern, between which those relationships will be described. As such the Role is a required association in a PatternDefinition. Semantically, a Role is a 'slot' that is required to be fulfilled for an instance of its parent PatternDefinition to exist."

In this specification measurements of Detection Pattern Occurrences rely on Roles in the following ways:

- Some patterns rely on roles that model values and threshold values. For example, in the CWE 1049 Detection Pattern, one Occurrence exists when the number of data queries (**CWE-1049-roles-numberOfDataQueries)** exceeds the number of data queries threshold value (**CWE-1049-roles-numberOfDataQueriesThresholdValue)**. Therefore, to correct this Weakness, the Occurrence Gap Size between these two values must be closed. In these cases (enumerated in normative Sub-clause 49.9), the Remediation Effort is estimated by multiplying a constant by the Gap Size.
- Contextual information collection relies on the implementation of these Roles.

6.7 Detection Pattern Comments

Some Detection Pattern definitions contain in the Comment pattern section the following term: (SPMS:Definitions::PatternSection). In the Automated Source Code Quality Measure specifications these comments indicate shared Detection Patterns between two or more ASCQM measures. For example, ASCSM-CWE-120-comment and ASCQM Check Index of Array Access-comment state that "Measure element contributes to Security and Reliability". Information in such comments is used to avoid duplicate counting of Remediation Effort in the overall Technical Debt score for a single Occurrence of CWE-120 that will appear in the calculations of both ASCSM and ASCRM.

6.8 Adherence to ASCMM, ASCRM, ASCSM, and ASCPEM Specifications

This specification refers to the ASCMM, ASCRM, ASCSM, and ASCPEM specifications via OCL operations relying on SPMS specifications:

- Detection Pattern Occurrences are identified by; **<pattern>.A_instanceOf_PatternInstance::PatternInstance()**. E.g., with *CWE-1075*: **ASCMM: ASCMMLibrary::ASCSM-CWE-1075.A_instanceOf_PatternInstance::PatternInstance()**

- Languages of code elements implementing the Detection Pattern Occurrence are identified by; **<pattern>.A_instanceOf_PatternInstance::PatternInstance().fulfillments().fulfilledBy().source().language()**. E.g., with *ASCSM-CWE-1075*: **ASCMM:ASCMMLibrary::ASCMM-CWE-1075.A_instanceOf_PatternInstance::PatternInstance().fulfillments().fulfilledBy().source().language()**

- Code elements implementing the Detection Pattern Occurrence roles are identified by; **<role>.A_boundTo_Binding::Binding().fulfilledBy()**. E.g., with *ASCMM-CWE-1075-roles-controlFlowJumpStatement*: **ASCMM:ASCMMLibrary::ASCMM-CWE-1075-roles-controlFlowJumpStatement.A_boundTo_Binding::Binding().fulfilledBy()**

6.9 Contextual Measures

Contextual measures quantify structural attributes of the software environment in which a specific Detection Pattern Occurrence is embedded that can cause variation in the time required to correct its Detection Pattern. The structural attributes quantified in Contextual measures include complexity, Concentration, Evolution , Exposure, and Technological Diversity. Contextual measures are used in Technical Debt calculations to adjust the Remediation times for each Detection Pattern Occurrence based on the impact of these structural attributes on the effort to correct each specific Occurrence. In this specification, Contextual measures related to pattern Occurrences are used in the following ways:

- They can be used in analyzing, interpreting, and using Technical Debt scores in making decisions, prioritizing Remediations, allocating resources, benchmarking, modeling, and other uses for Technical Debt results. For instance, when prioritizing the Remediation of a specific Detection Pattern Occurrence, the context surrounding the Detection Pattern influences the assessment of:
  - the operational risk associated with not correcting the Detection Pattern,
  - the destabilization risk associated with correcting the Detection Pattern,
  - the opportunity to reduce costs by removing many Detection Pattern Occurrences at the same time, and
  - the organizational risk associated with the synchronization of different teams to handle complex Detection Pattern Occurrences involving different technologies.

- They can be used in computing an Adjustment Factor for the Remediation Effort of each Detection Pattern Occurrence that accounts for the impact of structural attributes of the software environment in which the Detection Pattern Occurrence resides. For instance, when correcting a Detection Pattern, the required effort is impacted by the complexity of the code elements in which the Detection Pattern is embedded, their connectedness to other code elements in the software, the number of languages the Detection Pattern's implementation, etc.

Therefore, along with the identifying Detection Pattern Occurrences, the measurement of Technical Debt will include for each Occurrence the following measures:

- **Complexity—**of code elements, measured by the Effort Complexity, as defined in the Automated Enhancement Points (AEP) specification.
- **Exposure—**of the effects of the Detection Pattern Occurrence to the rest of the software system. Based on the extent of propagation, correcting the Occurrence could involve direct references to code elements (measured as the code elements' number of distinct direct

callers), or indirect references (measured as the number of distinct call paths leading to the code elements).
- **Technological Diversity—**the number of languages in which code elements composing a single Detection Pattern are instantiated.
- **Concentration—**total number of Detection Pattern Occurrences across all Weakness types within a single unit of code (e.g., class, module, component, subroutine, etc.).
- **Evolution —**changes and evolution both of code elements in the Detection Pattern Occurrence and of code elements constituting the immediate software environment within which the Occurrence is embedded.

In the context of Weaknesses which rely on roles that model values and threshold values that are not to be exceeded, the Gap Size for each Occurrence of a Weakness shall be collected and measured as the difference between the values and the threshold values.

These measures are included in the specification for Technical Debt to provide standard measures for use in interpreting Technical Debt information. Although organizations may develop their own interpretive measures, the use of these interpretive measures relieves an organization from having to develop its own proprietary adjustment formulas and provides standards for benchmarking adjusted values of Technical Debt. Expected benefits from using Contextual measures include the following:

- **Complexity**—identification of situations where correcting Technical Debt Items can lead to excessive effort and cost due to the complexity of the software in a Detection Pattern Occurrence is embedded.
- **Exposure**—identification of situations where the Remediation of Technical Debt Items can lead to excessive effort, cost, and unwanted side effects because the Detection Pattern Occurrence is coupled to an excessive number of code elements throughout the software system.
- **Technological Diversity—**identification of situations where effort and cost could rise because of the need to coordinate multiple individuals or teams with different language skills and knowledge of different system components.
- **Concentration**—identification of situations where corrective effort and cost could be reduced because the effort to understand, correct, and test fixes can be amortized across Technical Debt Items.
- **Evolution** — identification of situations where effort and cost can be reduced by incorporating the Remediation of Detection Pattern Occurrences into ongoing changes and evolution of the software in which the Occurrences are embedded.

6.10 Contextual Technical Debt Measure (CTDM)

Some organizations may want to customize the Automated Technical Debt Measure (ATDM) calculation to reflect local conditions or practices, thus turning a generic benchmarking indicator into a localized management indicator to help guide software development and maintenance decisions.

Such customizations may:

- exclude some Weaknesses from the calculation,
- adjust the default values for Unadjusted Remediation Effort,
- adapt the Adjustment Factor formula, by including or excluding contributing factors,
- adapt the formulae of the factors contributing to the Adjustment Factor.

These adjustments can be made for either the entire organization or for individual applications. Customized calculations shall be designated as a Contextual Technical Debt Measures (CTDM) to

distinguish them from the standard calculation (ATDM) which can be used for benchmarking with other organizations or datasets.

Informative Clause 9 provides some illustration and rationale behind CTDM customizations.

7 Automated Technical Debt Measure specification (normative)

7.1 Computing Process Overview

7.1.1 Automated Technical Debt Measure (ATDM)

Automated Technical Debt Measures (ATDM) shall be calculated through the following process:

1. Collect source code for one or two revisions of the software.

2. Generate the Application Model for the available revision(s), taking care of the evolveTo/evolveFrom relationships between code elements when there are two revisions.

3. Detect Occurrences of the Detection Patterns enumerated in the ASCQM standard.

4. Collect Contextual Information for each Detection Pattern Occurrence, i.e., Complexity, Exposure, Technological Diversity, Concentration, and Gap Size. Collect Evolution information when two revisions of the software were processed in steps 1, 2, and 3.
   a) Complexity is the Effort Complexity from the Automated Enhancement Points (*AEP*) standard.
   b) Exposure is the call graph branching factor, while Direct Exposure is the number of callers.
   c) Technological Diversity is the number of programming languages in which the code elements instantiating a single Detection Pattern are written.
   d) Concentration is the number of Detection Pattern Occurrences in a code element.
   e) Evolution requires determining whether a Detection Pattern Occurrence or code elements in the software environment where the Occurrence is embedded have been added, deleted, or updated between revisions of the software.
   f) Occurrence Gap Size, when the Detection Pattern incorporates roles that establish threshold values that are not to be exceeded.

5. Compute an Adjustment Factor for each Occurrence as the product of three Contextual measures (Complexity, Technological Diversity, and Concentration) from step 4:
   a) Complexity Overhead Average is computed as an average across the implementations of the Detection Pattern roles of their Complexity Overhead, measured as a ratio of the complexity from step 4.a divided by the lowest complexity value the implementations could have had (i.e., complexity as defined and calculated in the Automated Enhancement Points (*AEP*) standard).
   b) Technological Diversity for each Detection Pattern Occurrence is used as defined in step 4.c.
   c) Concentration Sharing Opportunity Average is computed as an average of the Concentration Sharing Opportunities across implementations of the Detection Pattern roles. Concentration Sharing Opportunity for each Detection Pattern role is measured as the inverse of the Concentration value from step 4.d.
   d) Exposure and Evolution are not used in the Adjustment Factor. Even so, the Exposure Overhead Average can be computed as an average of the Exposure Overheads across the implementations of Detection Pattern roles. The Exposure Overhead for a Detection Pattern role is measured as a logarithmic transformation of the Exposure

value from step 4.b. The Direct Exposure Overhead Average can also be computed as an average of the Direct Exposure values from step 4.b aggregated across implementations of Detection Pattern roles. Exposure and Evolution are valuable measures for managing technical debt (setting priorities, mitigating risk, etc.) and to build a localized Contextual Technical Debt Measure.

6. Multiply the Adjustment Factor from step 5 by the Unadjusted Remediation Effort to get the Adjusted Remediation Effort for each Detection Pattern Occurrence.

7. For each Detection Pattern, sum the Adjusted Remediation Efforts from step 6 across all Occurrences of the Detection Pattern to calculate the Occurrence Remediation Effort for the Detection Pattern

8. For each Weakness, sum the Occurrence Remediation Efforts from step 7 for all the Detection Patterns associated with the Weakness to calculate the Weakness Remediation Effort.

9. For each of the four Quality Characteristics, sum the Detection Pattern Remediation Efforts from step 7 for Detection Patterns detecting Weaknesses associated with that characteristic (*ASCMM*, *ASCRM*, *ASCPEM*, or *ASCSM)* to compute the Quality Characteristic Remediation Effort (MREM, PEREM, RREM, or SREM).

10. Sum all the Detection Pattern Remediation Efforts from step 7 to compute the Automated Technical Debt Measure (*ATDM*) for an application.

11. Sum Occurrence Remediation Efforts from step 6 for all Occurrences within a specified range of Contextual measures to build distributions of ATDM scores for the specified range.

Note on 8., 9., 10., and 11.:

1. Detection Patterns can help detect more than one Weakness,
2. Detection Patterns can be functionally included in other Detection Patterns (in *ASCQM* 1.0, this happens only once),
3. Weaknesses can be organized in optional parent-child relationships,
4. Weaknesses can be shared between quality characteristics, the sums must not deduplicate contributions to the total remediation effort.

Figures 2 and 3 visually summarize the computation formulae. They are provided for illustration and clarity purposes. However, they do not contain all the normative measure elements detailed in this clause.

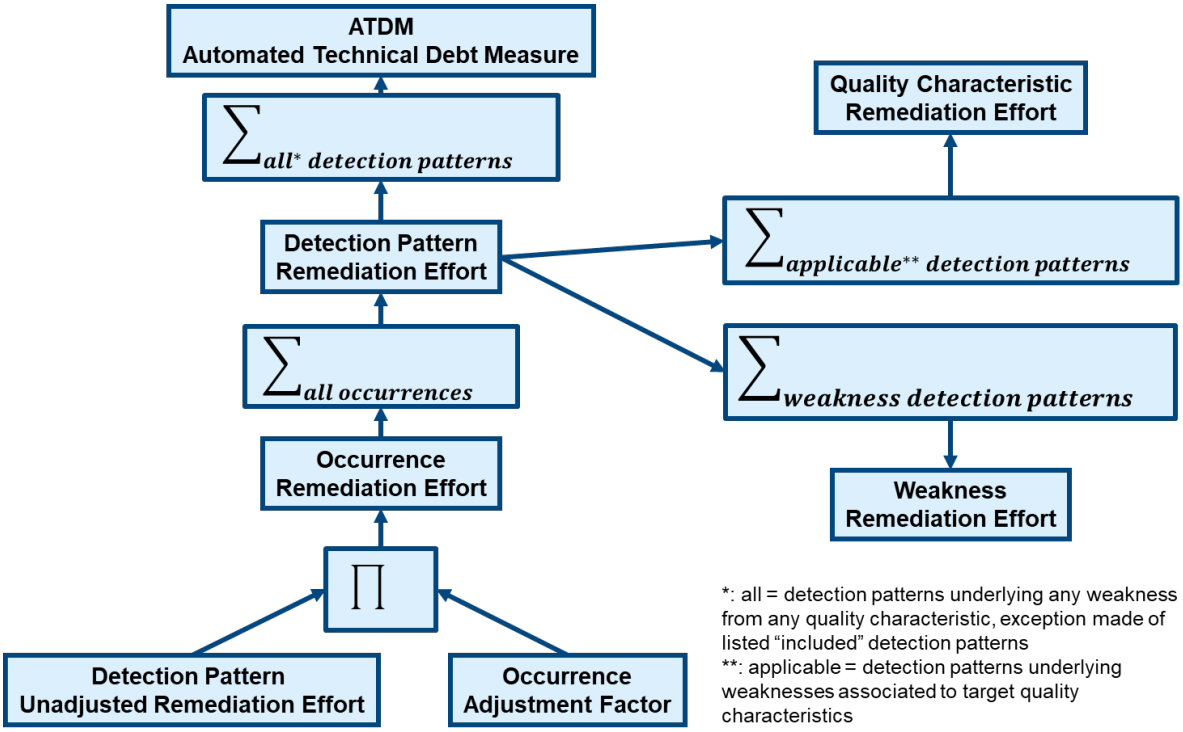

Figure 2. Illustration of the ATDM computation formula

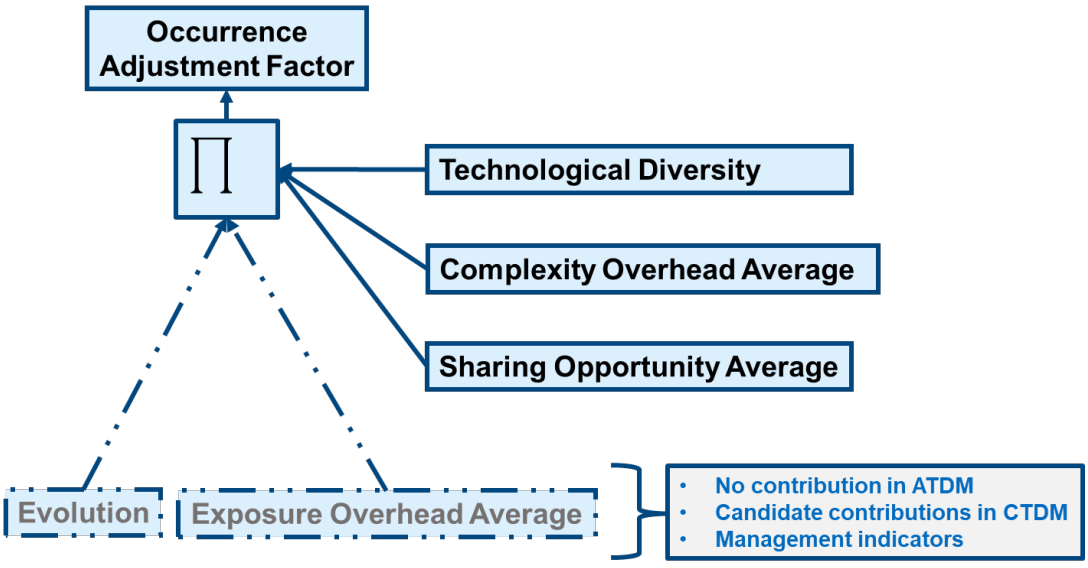

Figure 3. Illustration of Adjustment Factor computation formula

7.1.2 Contextual Technical Debt Measure (CTDM)

The process to follow in computing CTDM shall be identical to that for ATDM except for the following steps:

3. Detect Occurrences of selected Detection Patterns.
4. Collect contextual information in accordance to 5. Below.
5. Compute a custom Adjustment Factor.

6. Use custom Unadjusted Remediation Effort values.
7.-10. Sum Pattern and Weakness Remediation Efforts only for the selected Weaknesses.

7.2 Application Model

7.2.1 Overview

The calculation of the Automated Technical Debt Measure (*ATDM*) shall be performed:

- either on one revision of the software, which is called "*ToRevision*"
- or between two revisions of the software, which are called "*FromRevision*" and "*ToRevision*", with "*ToRevision*" being the more recent of the two revisions.

Each available revision shall be analyzed to create an Application Model of the software. The Application Model shall be composed of

- computational objects in the source code and their relationships, as defined in the *KDM* standard,
- Occurrences of Detection Patterns, including the binding information to the computational objects and relationships.

When both "*FromRevision*" and "T*oRevision*" revisions are available, the *evolvedTo/evolvedFrom* relationship shall be identified for all computational elements (i.e., to identify when code elements in "*FromRevision*" revision are also found in the "*ToRevision*" revision, and shall be identified as either an evolved version of the computational object, or an unchanged version) as presented in SMM Clause 17.1.

7.2.2 Representation in SMM of the revision(s)

SMM enables the following modeling:

- One **smm:Observation** of collected revision(s) so that the base Application Model shall contain all required items. **<measureElement xmi:type="smm:Observation" xmi:id="softwareMeasurementScope" name="softwareMeasurementScope" shortDescription="Observation of the Application Model which contains code elements from the final revision (and from the initial revision if available)." scopes="toRevisionMeasurementScope fromRevisionMeasurementScope"/>**
- One **smm:ObservationScope** in this **smm:Observation** for each revision shall be used to identify items from each revision.

7.2.3 Measure specifications

To handle the latest revision when two revisions are delivered, the analysis shall establish the following scope related entities:

- An **smm:ObservationScope <measureElement xmi:id="toRevisionMeasurementScope" xmi:type="smm:ObservationScope" name="toRevisionMeasurementScope" class="MOF::Element" shortDescription="Subset of the Application Model which contains code elements from the initial revision. Code elements are related to code elements from the final revision by evolvedTo/evolvedFrom relationships." />**

- An **smm:OCLOperation** to easily identify a code element from the **smm:ObservationScope <measureElement xmi:type="smm:OCLOperation" xmi:id="isInLatestRevision" name="isInLatestRevision" context="kdm:Core::Element" body="(toRevisionMeasurementScope()->includes(self))"/>**

To handle the previous revision when two revisions are delivered, the analysis shall establish the following scope related entities:

- A second **smm:ObservationScope** **<measureElement xmi:id="fromRevisionMeasurementScope" xmi:type="smm:ObservationScope" name="fromRevisionMeasurementScope" class="MOF::Element" shortDescription="Subset of the Application Model which contains code elements from the final revision. Code elements are related to code elements from the initial revision by evolvedTo/evolvedFrom relationships." />**
- A second **smm:OCLOperation** to easily identify a code element from the **smm:ObservationScope** **<measureElement xmi:type="smm:OCLOperation" xmi:id="isInPreviousRevision" name="isInPreviousRevision" context="kdm:Core::Element" body="(fromRevisionMeasurementScope()->includes(self))"/>**

7.3 Quantification of Remediation Effort at the Detection Pattern Occurrence Level

This sub-clause describes the steps that shall be used to compute the Remediation Effort measures of a single Detection Pattern Occurrence (Technical Debt Item) in a specific revision of the software.

For each Detection Pattern Occurrence in each revision, the effort (coding, unit/non-regression testing adaptation) to correct the Detection Pattern Occurrence shall be computed as a calculation conforming to the following process.

1) identify a Detection Pattern Occurrence
2) get Unadjusted Remediation Effort from Table 5
3) collect Contextual information
4) compute Adjustment Factor
5) compute Adjusted Remediation Effort

7.3.1 Occurrence identification

For each Detection Pattern, identify each individual Detection Pattern Occurrence through an **smm:Scope** relying on an **smm:Operation** to use as a scope recognizer. These items are demonstrated with the ASCQM Check Index of Array Access ASCQM Check Index of Array Access Detection Pattern as follows:

- an **smm:Scope**. **<measureElement xmi:type="smm:Scope" xmi:id="id.sfgd.34.scope" name="Occurrence Scope of ASCQM Check Index of Array Access" category="id.cat.277 id.cat.278" measures="id.sfgd.34.occurrence.count" class="spms:Observations::PatternInstance" recognizer="id.sfgd.34.recognizer"/>ASCQM Check Index of Array AccessASCQM Check Index of Array AccessASCQM Check Index of Array Access**
- defined by an OCL **smm:Operation** **<measureElement xmi:type="smm:Operation" xmi:id="id.sfgd.34.recognizer" name="Occurrence Scope Recognizer of ASCQM Check Index of Array Access" category="id.cat.277 id.cat.278" body="ascqm:id.sfgd.34.A_instanceOf_PatternInstance::PatternInstance()" language="OCL"/>ASCQM Check Index of Array AccessASCQM Check Index of Array AccessASCQM Check Index of Array Access**

Note that the key prefix is consistent with the key defined in *ASCQM* standard.

7.3.2 Measure specification

An **smm:Scope** measure (whose key is the Detection Pattern key with a '*.scope*' suffix) and its **smm:Operation** recognizer (whose key is the Detection Pattern key with an '*.recognizer*' suffix) shall be defined for each Detection Pattern from *ASCQM* standard, as illustrated with the ASCQM Check Index of Array Access Detection Pattern above.

7.3.3 Acquiring Unadjusted Remediation Effort Values

This sub-clause describes the steps that shall be used to get the Unadjusted Remediation Effort measure for all Occurrences (Technical Debt Items) of a specific Detection Pattern in each revision of the software, unadjusted by the Contextual information associated with the Detection Pattern Occurrence. For each Detection Pattern Occurrence in each revision, the unadjusted effort (coding, unit/non-regression testing adaptation) to remove the Detection Pattern Occurrence shall be determined as described below.

The Unadjusted Remediation Effort shall be modeled as an **smm:DirectMeasure** using an **smm:Operation** relying on a formula which uses a parameter to handle the Unadjusted Remediation Effort amount in minutes. These rules are demonstrated with the ASCQM Check Index of Array Access pattern as follows:

- an **smm:DirectMeasure**
  <measureElement xmi:type="smm:DirectMeasure" xmi:id="id.sfgd.34.unadjusted_remediation_effort" name="Occurrence Unadjusted Remediation Effort of ASCQM Check Index of Array Access" unit="effort(minutes)" scope="toRevisionMeasurementScope" trait="RemediationEffortEstimating" category="id.cat.277 id.cat.278" shortDescription="Effort to remove one occurrence of ASCQM Check Index of Array Access (in simplest context)" operation="id.sfgd.34.unadjusted_remediation_effort_value" baseMeasure1From="id.sfgd.34.remediation_effort_to_id.sfgd.34.unadjusted_remediation_effort"/>
- defined by an OCL **smm:Operation**
  <measureElement xmi:type="smm:Operation" xmi:id="id.sfgd.34.unadjusted_remediation_effort_value" name="Occurrence Unadjusted Remediation Effort Value of ASCQM Check Index of Array Access" trait="RemediationEffortEstimating" category="id.cat.277 id.cat.278" body="Real { id.sfgd.34.unadjusted_remediation_effort_value_occurrence_removal_effort_in_minutes = 46}" language="OCL"/>

The Unadjusted Remediation Value for an Occurrence of a Detection Pattern can be found in Table 5 in the row for the Weakness identified by this instantiation of the Detection Pattern.

7.3.4 Measure specifications

An **smm:DirectMeasure** measure (whose key is the Detection Pattern key with an '*.unadjusted_remediation_effort*' suffix) and its **smm:Operation** (whose key is the Detection Pattern key with a '*.unadjusted_remediation_effort_value'* suffix) shall be defined for each Detection Pattern from *ASCQM* standard, as illustrated with the ASCQM Check Index of Array Access pattern above. The Unadjusted Remediation Effort values are listed in Sub-clause 7.17.

7.4 Contextual measures of Detection Pattern Occurrences

This sub-clause describes the steps that shall be used to compute Contextual measures that can be applied to each individual Detection Pattern Occurrence. These Contextual measures are integral parts of the calculation of Technical Debt, via the Adjustment Factor detailed in Sub-clause 7.1 (5). These measures can also be used in analyzing, interpreting, and using Technical Debt values for making decisions, benchmarking, modeling, and other uses.

The measurement process shall include three sets of scopes:

- the code elements from the role implementations of each Detection Pattern Occurrence
- the parent Artifact, as defined in *AEP* standard, from the code elements from the role implementations of each Detection Pattern Occurrence
- the languages in which code elements were implemented, from the role implementations of each Detection Pattern Occurrence

Then, the measurement process shall compute the following Contextual measures:

- Technological Diversity, using the language-related scopes,
- Complexity, using the parent Artifact scopes,
- Exposure, Direct Exposure, Concentration, and Evolution, using the code-elements-related scopes.

Last, when applicable, the measurement process shall compute the Occurrence Gap Size.

7.4.1 Detection Pattern Occurrence Implementation Code Elements

An **smm:Scope** (named as the role name with a '**_code_elements** ' suffix), and its recognizer **smm:Operation** (named as the role name with a '**_code_elements** *_recognizer'* suffix) shall be defined for each applicable Role (listed below) in Detection Patterns from ASCQM standard. *ASCQM Check Index of Array Access role* **PathFromDeclarationStatementToUseAsAnIndexStatement** will be used in the examples below:

- an **smm:Scope**
  <measureElement xmi:type="smm:Scope"
  xmi:id="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_code_elements" name="ASCQM Check Index of Array Access PathFromDeclarationStatementToUseAsAnIndexStatement Code Elements"
  category="id.cat.277 id.cat.278"
  operation="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_code_elements_recognizer" class="kdm:Code::AbstractCodeElement"/>
- relying on an **smm:Operation**
  <measureElement xmi:type="smm:Operation"
  xmi:id="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_code_elements_recognizer" name="ASCQM Check Index of Array Access PathFromDeclarationStatementToUseAsAnIndexStatement Code Elements Recognizer"
  category="id.cat.277 id.cat.278"
  body="ascqm:id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_code_elements_recognizer.A_boundTo_Binding::Binding().fulfilledBy()" language="OCL"/>

7.4.2 Measure specifications

An **smm:Scope** measure (named as the role key with a '**_code_elements**' suffix) and its **smm:Operation** recognizer (whose key is the Detection Pattern with a '**_code_elements_recognizer** ' suffix) shall be defined for each applicable role from source code pattern from *ASCQM* standard, as illustrated with the *ASCQM Check Index of Array Access role* **PathFromDeclarationStatementToUseAsAnIndexStatement** above.

Applicable roles are:

- ASCQM Ban Allocation of Memory with Null Size MemoryAllocationCall Code Elements
- ASCQM Ban Assignment Operation Inside Logic Blocks AssignmentExpression Code Elements
- ASCQM Ban Assignment Operation Inside Logic Blocks LogicBlock Code Elements
- ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size MemoryAllocationCall Code Elements
- ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation BitwiseOperation Code Elements
- ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation MemoryAllocationCall Code Elements
- ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value LengthComputation Code Elements
- ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value MemoryAllocationCall Code Elements
- ASCQM Ban Comma Operator from Delete Statement CommaStatement Code Elements
- ASCQM Ban Comma Operator from Delete Statement DeleteStatement Code Elements
- ASCQM Ban Comparison Expression Outside Logic Blocks ComparisonExpression Code Elements
- ASCQM Ban Control Flow Transfer ControlFlowJumpStatement Code Elements
- ASCQM Ban Conversion References to Child Class Class Code Elements
- ASCQM Ban Conversion References to Child Class ParentClass Code Elements
- ASCQM Ban Conversion References to Child Class TypeConversion Code Elements
- ASCQM Ban Creation of Lock on Inappropriate Object Type LockingAcquisitionStatement Code Elements
- ASCQM Ban Creation of Lock on Inappropriate Object Type ObjectDeclaration Code Elements
- ASCQM Ban Creation of Lock on Non-Final Object LockingAcquisitionStatement Code Elements
- ASCQM Ban Creation of Lock on Non-Final Object NonFinalObjectDeclaration Code Elements
- ASCQM Ban Creation of Lock on Private Non-Static Object to Access Private Static Data DataAccess Code Elements
- ASCQM Ban Creation of Lock on Private Non-Static Object to Access Private Static Data PrivateNonStaticLock Code Elements
- ASCQM Ban Creation of Lock on Private Non-Static Object to Access Private Static Data PrivateStaticData Code Elements
- ASCQM Ban Delete of VOID Pointer DeclarationStatement Code Elements
- ASCQM Ban Delete of VOID Pointer ReleaseStatement Code Elements
- ASCQM Ban Double Free on Pointers FirstPointerReleaseStatement Code Elements
- ASCQM Ban Double Free on Pointers PathToPointerReleaseFromPointerRelease Code Elements

- ASCQM Ban Double Free on Pointers SecondPointerReleaseStatement Code Elements
- ASCQM Ban Double Release of Resource FirstResourceReleaseStatement Code Elements
- ASCQM Ban Double Release of Resource PathToResourceReleaseFromResourceRelease Code Elements
- ASCQM Ban Double Release of Resource SecondResourceReleaseStatement Code Elements
- ASCQM Ban Empty Exception Block CatchBlock Code Elements
- ASCQM Ban Exception Definition without Ever Throwing It Exception Code Elements
- ASCQM Ban Exception Definition without Ever Throwing It FunctionProcedureOrMethod Code Elements
- ASCQM Ban Excessive Complexity of Data Resource Access Query Code Elements
- ASCQM Ban Excessive Number of Children Class Code Elements
- ASCQM Ban Excessive Number of Concrete Implementations to Inherit From Class Code Elements
- ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code FunctionProcedureOrMethod Code Elements
- ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure Function Code Elements
- ASCQM Ban Excessive Number of Index on Columns of Large Tables Code Elements
- ASCQM Ban Excessive Number of Inheritance Levels Class Code Elements
- ASCQM Ban Excessive Size of Index on Columns of Large Tables Code Elements
- ASCQM Ban Expensive Operations in Loops Loop Code Elements
- ASCQM Ban Expensive Operations in Loops ResourceConsummingStatement Code Elements
- ASCQM Ban File Creation with Default Permissions FileCreationStatement Code Elements
- ASCQM Ban File Creation with Default Permissions Permission Code Elements
- ASCQM Ban Free Operation on Pointer Received as Parameter ReleaseStatement Code Elements
- ASCQM Ban Free Operation on Pointer Received as Parameter Signature Code Elements
- ASCQM Ban Hard-Coded Literals used to Connect to Resource InitializationStatement Code Elements
- ASCQM Ban Hard-Coded Literals used to Connect to Resource ResourceAccessStatement Code Elements
- ASCQM Ban Hard-Coded Literals used to Initialize Variables InitializationStatement Code Elements
- ASCQM Ban Incompatible Lock Acquisition Sequences LockAcquisitionSequence Code Elements
- ASCQM Ban Incompatible Lock Acquisition Sequences ReverseLockAcquisitionSequence Code Elements
- ASCQM Ban Incorrect Float Number Comparison FloatEqualityComparisonExpression Code Elements
- ASCQM Ban Incorrect Joint Comparison JointComparisonExpression Code Elements
- ASCQM Ban Incorrect Numeric Conversion of Return Value CallStatement Code Elements
- ASCQM Ban Incorrect Numeric Conversion of Return Value FunctionMethodOrProcedure Code Elements
- ASCQM Ban Incorrect Numeric Conversion of Return Value TargetDataType Code Elements

- ASCQM Ban Incorrect Numeric Conversion of Return Value VariableDataType Code Elements
- ASCQM Ban Incorrect Numeric Implicit Conversion Data Code Elements
- ASCQM Ban Incorrect Numeric Implicit Conversion TargetDataType Code Elements
- ASCQM Ban Incorrect Numeric Implicit Conversion Variable Code Elements
- ASCQM Ban Incorrect Numeric Implicit Conversion VariableAssignmentStatement Code Elements
- ASCQM Ban Incorrect Numeric Implicit Conversion VariableDataType Code Elements
- ASCQM Ban Incorrect Object Comparison ObjectEqualityComparisonExpression Code Elements
- ASCQM Ban Incorrect String Comparison StringEqualityComparisonExpression Code Elements
- ASCQM Ban Incorrect Synchronization Mechanisms IncorrectSynchronizationPrimitiveCall Code Elements
- ASCQM Ban Incorrect Type Conversion Data Code Elements
- ASCQM Ban Incorrect Type Conversion TargetDataType Code Elements
- ASCQM Ban Incorrect Type Conversion Variable Code Elements
- ASCQM Ban Incorrect Type Conversion VariableAssignmentStatement Code Elements
- ASCQM Ban Incorrect Type Conversion VariableDataType Code Elements
- ASCQM Ban Incremental Creation of Immutable Data StringConcatenationStatement Code Elements
- ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities InputAcquisitionCall Code Elements
- ASCQM Ban Logical Dead Code FunctionProcedureOrMethod Code Elements
- ASCQM Ban Logical Dead Code Statement Code Elements
- ASCQM Ban Logical Operation with a Constant Operand ComparisonExpression Code Elements
- ASCQM Ban Loop Value Update within Incremental and Decremental Loop LoopVariable Code Elements
- ASCQM Ban Loop Value Update within Incremental and Decremental Loop LoopVariableUpdateStatement Code Elements
- ASCQM Ban Non-Final Static Data in Multi-Threaded Context Declaration Code Elements
- ASCQM Ban Non-Serializable Elements in Serializable Objects NonSerializableMember Code Elements
- ASCQM Ban Non-Serializable Elements in Serializable Objects SerializableClass Code Elements
- ASCQM Ban Not Operator on Non-Boolean Operand Of Comparison Operation ComparisonExpression Code Elements
- ASCQM Ban Not Operator on Operand Of Bitwise Operation BitwiseExpression Code Elements
- ASCQM Ban Reading and Writing the Same Variable Used as Assignment Value VariableAssignment Code Elements
- ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context ResourceAccessStatement Code Elements
- ASCQM Ban Return of Local Variable Address LocalVariable Code Elements
- ASCQM Ban Return of Local Variable Address Operation Code Elements
- ASCQM Ban Self Assignment SelfAssignmentStatement Code Elements
- ASCQM Ban Self Destruction DeleteThisStatement Code Elements

- ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock FirstLockAcquisitionStatement Code Elements
- ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock SecondLockAcquisitionStatement Code Elements
- ASCQM Ban Sleep Between Lock Acquisition and Release LockAcquisitionStatement Code Elements
- ASCQM Ban Sleep Between Lock Acquisition and Release LockReleaseStatement Code Elements
- ASCQM Ban Sleep Between Lock Acquisition and Release PathFromLockAcquisitionToLockRelease Code Elements
- ASCQM Ban Sleep Between Lock Acquisition and Release SleepStatement Code Elements
- ASCQM Ban Static Non-Final Data Element Outside Singleton StaticNonFinalVariableDeclaration Code Elements
- ASCQM Ban Storage of Local Variable Address in Global Variable GlobalVariable Code Elements
- ASCQM Ban Storage of Local Variable Address in Global Variable LocalVariable Code Elements
- ASCQM Ban Storage of Local Variable Address in Global Variable StorageStatement Code Elements
- ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities StringManipulationCall Code Elements
- ASCQM Ban Switch in Switch Statement NestedSwitch Code Elements
- ASCQM Ban Switch in Switch Statement ParentSwitch Code Elements
- ASCQM Ban Unintended Paths Callee Code Elements
- ASCQM Ban Unintended Paths Caller Code Elements
- ASCQM Ban Unintended Paths Relation Code Elements
- ASCQM Ban Unmodified Loop Variable Within Loop WhileLoop Code Elements
- ASCQM Ban Unreferenced Dead Code FunctionProcedureOrMethod Code Elements
- ASCQM Ban Usage of Data Elements from Other Classes Class Code Elements
- ASCQM Ban Usage of Data Elements from Other Classes OtherClass Code Elements
- ASCQM Ban Usage of Data Elements from Other Classes Reference Code Elements
- ASCQM Ban Use of Deprecated Libraries CallStatement Code Elements
- ASCQM Ban Use of Deprecated Libraries DeprecatedLibrary Code Elements
- ASCQM Ban Use of Expired Pointer PathToPointerAccessFromPointerRelease Code Elements
- ASCQM Ban Use of Expired Pointer PointerAccessStatement Code Elements
- ASCQM Ban Use of Expired Pointer PointerReleaseStatement Code Elements
- ASCQM Ban Use of Expired Resource PathToResourceAccessFromResourceRelease Code Elements
- ASCQM Ban Use of Expired Resource ResourceAccessStatement Code Elements
- ASCQM Ban Use of Expired Resource ResourceReleaseStatement Code Elements
- ASCQM Ban Use of Prohibited Low-Level Resource Management Functionality ResourceManagementPrimitiveCall Code Elements
- ASCQM Ban Use of Prohibited Low-Level Resource Management Functionality TechnologyStack Code Elements
- ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues ThreadControlPrimitiveCall Code Elements
- ASCQM Ban Useless Handling of Exceptions CatchBlock Code Elements

- ASCQM Ban Variable Increment or Decrement Operation in Operations using the Same Variable VariableAssignment Code Elements
- ASCQM Ban While TRUE Loop Without Path To Break WhileTrueLoop Code Elements
- ASCQM Catch Exceptions Exception Code Elements
- ASCQM Catch Exceptions Method Code Elements
- ASCQM Catch Exceptions MethodCall Code Elements
- ASCQM Check and Handle ZERO Value before Use as Divisor DivisionStatement Code Elements
- ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use Boolean Code Elements
- ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use Condition Code Elements
- ASCQM Check Index of Array Access ArrayAccessStatement Code Elements
- ASCQM Check Index of Array Access PathFromDeclarationStatementToUseAsAnIndexStatement Code Elements
- ASCQM Check Index of Array Access VariableDeclarationStatement Code Elements
- ASCQM Check Input of Memory Allocation Primitives MemoryAllocationCall Code Elements
- ASCQM Check Input of Memory Manipulation Primitives MemoryManipulationCall Code Elements
- ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities StringManipulationCall Code Elements
- ASCQM Check NULL Pointer Value before Use EvaluationStatement Code Elements
- ASCQM Check Offset used in Pointer Arithmetic ArithmeticExpression Code Elements
- ASCQM Check Offset used in Pointer Arithmetic EvaluationStatement Code Elements
- ASCQM Check Return Value of Resource Operations Immediately CallToTheOperation Code Elements
- ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context InitializationStatement Code Elements
- ASCQM Handle Return Value of Must Check Operations CallToTheOperation Code Elements
- ASCQM Handle Return Value of Resource Operations CallToTheOperation Code Elements
- ASCQM Implement Copy Constructor for Class with Pointer Resource Class Code Elements
- ASCQM Implement Copy Constructor for Class with Pointer Resource Pointer Code Elements
- ASCQM Implement Correct Object Comparison Operations Class Code Elements
- ASCQM Implement Index Required by Query on Large Tables Column Code Elements
- ASCQM Implement Index Required by Query on Large Tables Query Code Elements
- ASCQM Implement Index Required by Query on Large Tables Code Elements
- ASCQM Implement Required Operations for Manual Resource Management ObjectDeclaration Code Elements
- ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor Class Code Elements
- ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor ParentClass Code Elements
- ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor ParentVirtualDestructor Code Elements

- ASCQM Implement Virtual Destructor for Classes with Virtual Methods Class Code Elements
- ASCQM Implement Virtual Destructor for Classes with Virtual Methods VirtualMethod Code Elements
- ASCQM Implement Virtual Destructor for Parent Classes Class Code Elements
- ASCQM Implement Virtual Destructor for Parent Classes ParentClass Code Elements
- ASCQM Initialize Pointers before Use PathToPointerAccessFromPointerDeclaration Code Elements
- ASCQM Initialize Pointers before Use PointerAccessStatement Code Elements
- ASCQM Initialize Pointers before Use PointerDeclarationStatement Code Elements
- ASCQM Initialize Resource before Use PathToResourceAccessFromResourceDeclaration Code Elements
- ASCQM Initialize Resource before Use ResourceAccessStatement Code Elements
- ASCQM Initialize Resource before Use ResourceDeclarationStatement Code Elements
- ASCQM Initialize Variables before Use PathToVariableAccessFromVariableDeclaration Code Elements
- ASCQM Initialize Variables before Use VariableAccessStatement Code Elements
- ASCQM Initialize Variables before Use VariableDeclarationStatement Code Elements
- ASCQM Initialize Variables PathFromVariableDeclaration Code Elements
- ASCQM Initialize Variables VariableDeclarationStatement Code Elements
- ASCQM Limit Algorithmic Complexity via Cyclomatic Complexity Value FunctionProcedureOrMethod Code Elements
- ASCQM Limit Algorithmic Complexity via Essential Complexity Value FunctionProcedureOrMethod Code Elements
- ASCQM Limit Algorithmic Complexity via Module Design Complexity Value FunctionProcedureOrMethod Code Elements
- ASCQM Limit Number of Aggregated Non-Primitive Data Types Class Code Elements
- ASCQM Limit Number of Data Access FunctionProcedureOrMethod Code Elements
- ASCQM Limit Number of Outward Calls FunctionProcedureOrMethod Code Elements
- ASCQM Limit Number of Parameters FunctionProcedureOrMethod Code Elements
- ASCQM Limit Size of Operations Code FunctionProcedureOrMethod Code Elements
- ASCQM Limit Volume of Commented-Out Code FunctionProcedureOrMethod Code Elements
- ASCQM Limit Volume of Similar Code FunctionProcedureOrMethod1 Code Elements
- ASCQM Limit Volume of Similar Code FunctionProcedureOrMethod2 Code Elements
- ASCQM Log Caught Security Exceptions CatchStatement Code Elements
- ASCQM Log Caught Security Exceptions Method Code Elements
- ASCQM Log Caught Security Exceptions MethodCall Code Elements
- ASCQM Log Caught Security Exceptions SecurityException Code Elements
- ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls BlockingSynchronousCall Code Elements
- ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls TimeOutOption Code Elements
- ASCQM NULL Terminate Output Of String Manipulation Primitives StringManipulationCallStatement Code Elements
- ASCQM Release File Resource after Use in Class Class Code Elements
- ASCQM Release File Resource after Use in Class FileResourceOpenStatement Code Elements
- ASCQM Release File Resource after Use in Operation FileResourceOpenStatement Code Elements

- ASCQM Release File Resource after Use in Operation FunctionProcedureOrMethod Code Elements
- ASCQM Release File Resource after Use in Operation PathToExitWithoutFileResourceClose Code Elements
- ASCQM Release in Destructor Memory Allocated in Constructor MemoryAllocationStatement Code Elements
- ASCQM Release Lock After Use FunctionProcedureOrMethod Code Elements
- ASCQM Release Lock After Use LockAcquisitionStatement Code Elements
- ASCQM Release Lock After Use PathToExitWithoutLockRelease Code Elements
- ASCQM Release Memory After Use MemoryAllocationStatement Code Elements
- ASCQM Release Memory after Use with Correct Operation MemoryAllocationStatement Code Elements
- ASCQM Release Memory after Use with Correct Operation MemoryReleaseStatement Code Elements
- ASCQM Release Platform Resource after Use FunctionProcedureOrMethod Code Elements
- ASCQM Release Platform Resource after Use PathToExitWithoutResourceRelease Code Elements
- ASCQM Release Platform Resource after Use ResourceAllocationStatement Code Elements
- ASCQM Sanitize Stored Input used in User Output PathFromRetrievalStatementToUserDisplay Code Elements
- ASCQM Sanitize Stored Input used in User Output PathFromUserInputToStorageStatement Code Elements
- ASCQM Sanitize Stored Input used in User Output RetrievalStatement Code Elements
- ASCQM Sanitize Stored Input used in User Output StorageStatement Code Elements
- ASCQM Sanitize Stored Input used in User Output UserDisplay Code Elements
- ASCQM Sanitize Stored Input used in User Output UserInput Code Elements
- ASCQM Sanitize User Input used as Array Index ArrayAccessStatement Code Elements
- ASCQM Sanitize User Input used as Array Index PathFromUserInputToArrayAccess Code Elements
- ASCQM Sanitize User Input used as Array Index UserInput Code Elements
- ASCQM Sanitize User Input used as Pointer PathFromUserInputToPointerDereferencing Code Elements
- ASCQM Sanitize User Input used as Pointer PointerDereferencingStatement Code Elements
- ASCQM Sanitize User Input used as Pointer UserInput Code Elements
- ASCQM Sanitize User Input used as Serialized Object DeserializationStatement Code Elements
- ASCQM Sanitize User Input used as Serialized Object PathFromUserInputToDeserialization Code Elements
- ASCQM Sanitize User Input used as Serialized Object UserInput Code Elements
- ASCQM Sanitize User Input used as String Format FormatStatement Code Elements
- ASCQM Sanitize User Input used as String Format PathFromUserInputToFormatStatement Code Elements
- ASCQM Sanitize User Input used as String Format UserInput Code Elements
- ASCQM Sanitize User Input used in Document Manipulation Expression DocumentManipulationExpression Code Elements
- ASCQM Sanitize User Input used in Document Manipulation Expression PathFromUserInputToDocumentManipulation Code Elements

- ASCQM Sanitize User Input used in Document Manipulation Expression UserInput Code Elements
- ASCQM Sanitize User Input used in Document Navigation Expression DocumentNavigationEvaluationExpression Code Elements
- ASCQM Sanitize User Input used in Document Navigation Expression PathFromUserInputToDocumentNavigationEvaluation Code Elements
- ASCQM Sanitize User Input used in Document Navigation Expression UserInput Code Elements
- ASCQM Sanitize User Input used in Expression Language Statement ExpressionLanguageExpression Code Elements
- ASCQM Sanitize User Input used in Expression Language Statement TransformationSequence Code Elements
- ASCQM Sanitize User Input used in Expression Language Statement UserInput Code Elements
- ASCQM Sanitize User Input used in Loop Condition LoopConditionStatement Code Elements
- ASCQM Sanitize User Input used in Loop Condition PathFromUserInputToLoopCondition Code Elements
- ASCQM Sanitize User Input used in Loop Condition UserInput Code Elements
- ASCQM Sanitize User Input used in Path Manipulation PathFromUserInputToPathManipulation Code Elements
- ASCQM Sanitize User Input used in Path Manipulation PathManipulationStatement Code Elements
- ASCQM Sanitize User Input used in Path Manipulation UserInput Code Elements
- ASCQM Sanitize User Input used in SQL Access PathFromUserInputToSQLStatement Code Elements
- ASCQM Sanitize User Input used in SQL Access SQLStatement Code Elements
- ASCQM Sanitize User Input used in SQL Access UserInput Code Elements
- ASCQM Sanitize User Input used in System Command ExecuteRunTimeCommandStatement Code Elements
- ASCQM Sanitize User Input used in System Command PathFromUserInputToExecuteRunTimeCommand Code Elements
- ASCQM Sanitize User Input used in System Command UserInput Code Elements
- ASCQM Sanitize User Input used in User Output PathFromUserInputToUserDisplay Code Elements
- ASCQM Sanitize User Input used in User Output UserDisplay Code Elements
- ASCQM Sanitize User Input used in User Output UserInput Code Elements
- ASCQM Sanitize User Input used to access Directory Resources DirectoryAccessStatement Code Elements
- ASCQM Sanitize User Input used to access Directory Resources PathFromUserInputToExecuteRunTimeCommand Code Elements
- ASCQM Sanitize User Input used to access Directory Resources UserInput Code Elements
- ASCQM Secure Use of Unsafe XML Processing with Secure Parser XMLProcessingCall Code Elements
- ASCQM Secure XML Parsing with Secure Options DTDProcessingDisablingOption Code Elements
- ASCQM Secure XML Parsing with Secure Options XMLParsingCall Code Elements
- ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context InitializationStatement Code Elements

- ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context SingletonClass Code Elements
- ASCQM Use Break in Switch Statement ControlFlowBranch Code Elements
- ASCQM Use Break in Switch Statement Switch Code Elements
- ASCQM Use Default Case in Switch Statement Switch Code Elements

7.4.3 Detection Pattern Occurrence Parent Artifact Code Elements

An **smm:Scope** (named as the role name with a '*_parent_artifact*_**code_elements**' suffix), and its recognizer **smm:Operation** (named as the role name with a '*_parent_artifact*_**code_elements** *_recognizer'* suffix) shall be defined for each applicable Role (listed below) in Detection Patterns from *ASCQM* standard. *ASCQM Check Index of Array Access role* **PathFromDeclarationStatementToUseAsAnIndexStatement** will be used in the examples below:

- an **smm:Scope**
  <measureElement xmi:type="smm:Scope" xmi:id="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_parent_artifact_code_elements" name="ASCQM Check Index of Array Access PathFromDeclarationStatementToUseAsAnIndexStatement Code Elements from parent Artifact" category="id.cat.277 id.cat.278" operation="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_parent_artifact_code_elements_recognizer" class="aep::Artifact"/>
- relying on an **smm:Operation**
  <measureElement xmi:type="smm:Operation" xmi:id="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_parent_artifact_code_elements_recognizer" name="ASCQM Check Index of Array Access PathFromDeclarationStatementToUseAsAnIndexStatement Code Elements from parent Artifact Recognizer" category="id.cat.277 id.cat.278" body="ascqm:id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_parent_artifact_code_elements_recognizer.A_boundTo_Binding::Binding().fulfilledBy().closure(parent)->select(oclIsTypeOf(aep::Artifact))->notEmpty()" language="OCL"/>

7.4.4 Measure specifications

An **smm:Scope** measure (named as the role key with a '_code_elements' suffix) and its **smm:Operation** recognizer (whose key is the Detection Pattern with a '_code_elements_recognizer ' suffix) shall be defined for each applicable role from source code pattern from *ASCQM* standard, as illustrated with the ASCQM Check Index of Array Access role PathFromDeclarationStatementToUseAsAnIndexStatement above.

7.5 Technological Diversity

Technological Diversity is the number of distinct languages in which the code elements of a single Occurrence of a Detection Pattern are written. Technological Diversity shall be computed as a simple count applied to the Detection Pattern Occurrence implementation languages scopes.

E.g., with *ASCQM Check Index of Array Access*:

- an **smm:Counting** measure
  <measureElement xmi:type="smm:Counting" xmi:id="id.sfgd.34.technological_diversity" name="Occurrence Technological Diversity of ASCQM Check Index of Array Access"

unit="Integer" scope="id.sfgd.34.code_element_languages" trait="LanguageCounting" category="id.cat.277 id.cat.278" shortDescription="Occurrence Technological Diversity of ASCQM Check Index of Array Access (measured as the number of distinct languages)" baseMeasureFrom="id.sfgd.34.adjustment_factor_to_id.sfgd.34.technological_diversity"/>

7.5.1 Measure specifications

An **smm:Counting** measure (whose key is the Detection Pattern with a '**.technological_diversity** ' suffix) shall be defined for each source code pattern from *ASCQM standard*, as illustrated with the *ASCQM Check Index of Array Access* pattern above.

7.5.2 Occurrence implementation languages

The set of languages in which a single pattern Occurrence has been implemented shall be determined through the following process:

1. For each Detection Pattern Occurrence, list implementation code elements, regardless of the role,
2. For each code element, list the source region(s),
3. For each source region, collect the language attribute value.

An **smm:Scope** (whose key is the Detection Pattern name with a '**.code_element_languages** ' suffix), and its recognizer **smm:Operation** (whose key is the Detection Pattern name with a '. **code_element_languages**_*recognizer'* suffix) shall be defined for each Detection Pattern.

E.g., with *ASCQM Check Index of Array Access*:

- an **smm:Scope**
  <measureElement xmi:type="smm:Scope" xmi:id="id.sfgd.34.code_element_languages" name="Occurrence Code Element Languages of ASCQM Check Index of Array Access" category="id.cat.277 id.cat.278" class="MOF::Element" recognizer="id.sfgd.34.code_element_languages_recognizer"/>
- relying on an **smm:Operation**
  <measureElement xmi:type="smm:Operation" xmi:id="id.sfgd.34.code_element_languages_recognizer" name="Occurrence Code Element Languages of ASCQM Check Index of Array Access Recognizer" category="id.cat.277 id.cat.278" body="ascqm:id.sfgd.34.code_element_languages_recognizer.A_instanceOf_PatternInstance::PatternInstance().fulfillments().fulfilledBy().source().language()" language="OCL"/>

7.5.3 Measure specifications

An **smm:Scope** measure (whose key is the Detection Pattern key with a '**.code_element_languages**' suffix) and its **smm:Operation** recognizer (whose key is the Detection Pattern key with a '**.code_element_languages** _*recognizer'* suffix) shall be defined for each Detection Pattern from *ASCQM standard*, as illustrated with the *ASCQM Check Index of Array Access* Detection Pattern above.

7.6 Complexity

Complexity, or Effort Complexity, shall be measured as defined in the *Automated Enhancement Points* specification, via an **smm:NamedMeasure**.

<measureElement xmi:type="smm:NamedMeasure" xmi:id="ArtifactEffortComplexity" name="ArtifactEffortComplexity" unit="ImplementationPoint" scope="aep::Artifact"

trait="ImplementationComplexity" shortDescription="Code Element Effort Complexity according to AEP 1.0 specifications" formula="aep::ArtifactEffortComplexity" baseMeasure1From="id.sfgd.100.Table_complexity_overhead_to_ArtifactEffortComplexity id.sfgd.101.Table_complexity_overhead_to_ArtifactEffortComplexity id.sfgd.102.FunctionProcedureOrMethod_complexity_overhead_to_ArtifactEffortComplexity … " />

aep::Artifact is a subset of **kdm:code::ControlElement** and this measure will return non-null values for elements of this subset only.

To compute the Complexity overhead which contributes to the Adjustment Factor, the Low Complexity Effort value shall also be collected via a second **smm:NamedMeasure**. This is the lowest complexity value the implementation code elements could have had, considered to be the "best case scenario" for well-implemented code.

<measureElement xmi:type="smm:NamedMeasure" xmi:id="LowEffortComplexity" name="LowEffortComplexity" unit="ImplementationPoint" scope="aep::Artifact" trait="ImplementationComplexity" shortDescription="Code Element lowest Effort Complexity value according to AEP 1.0 specifications" formula="aep::wLowEC" baseMeasure2From="id.sfgd.100.Table_complexity_overhead_to_LowEffortComplexity … " />

For each implementation role, the ratio of the two above values defines a complexity overhead, via an **smm:RatioMeasure**.

E.g., with *ASCQM Check Index of Array Access role* **PathFromDeclarationStatementToUseAsAnIndexStatement**:

<measureElement xmi:type="smm:RatioMeasure" xmi:id="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_complexity_overhead" name="ASCQM Check Index of Array Access PathFromDeclarationStatementToUseAsAnIndexStatement Code Elements from parent Artifact" unit="Real" scope="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_parent_artifact_code_elements" trait="ComplexityEstimating" category="id.cat.277 id.cat.278" shortDescription="ASCQM Check Index of Array Access PathFromDeclarationStatementToUseAsAnIndexStatement Code Elements from parent Artifact of code elements (measured as their Effort Complexity divided by the minimal Effort Complexity they could have)" baseMeasure1To="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_complexity_overhead_to_ArtifactEffortComplexity" baseMeasure2To="id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_complexity_overhead_to_LowEffortComplexity" baseMeasureFrom="id.sfgd.34.complexity_overhead_average_to_id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_complexity_overhead"/>

Measure specifications

An **smm:RatioMeasure** measure (named as the role key with a '_complexity overhead' suffix) shall be defined for each implementation role from *ASCQM standard* patterns, as illustrated with the *ASCQM Check Index of Array Access role* **PathFromDeclarationStatementToUseAsAnIndexStatement** above.

7.7 Exposure and Direct Exposure

Exposure and Direct Exposure shall be measured for all Detection Pattern Occurrences, respectively as the number of distinct call graph to and the number of direct callers of the code elements from the implementation of the Detection Pattern roles.

For each Detection Pattern Role, the associated **smm:Scope** (named as the role name with a '*_code_elements*' suffix), and its recognizer **smm:Operation** (named as the role name with a '*_code_elements_recognizer*' suffix) will be reused in the current process.

7.7.1 User input Exposure considerations

In case of a Detection Pattern relying on user input, the number of distinct callers and call paths shall be 0, but the Exposure is virtually infinite as the Detection Pattern is directly exposed to the outside world. From the security standpoint, the probability for an event (a malevolent use of the entry point into the system) to occur is “1”. This shall be considered when using Exposure to manage decisions or outcomes related to Technical Debt.

The affected patterns are:

- ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities
- ASCQM Check Input of Memory Allocation Primitives
- ASCQM Check Input of Memory Manipulation Primitives
- ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities
- ASCQM Sanitize Stored Input used in User Output
- ASCQM Sanitize User Input used as Array Index
- ASCQM Sanitize User Input used as Pointer
- ASCQM Sanitize User Input used as Serialized Object
- ASCQM Sanitize User Input used as String Format
- ASCQM Sanitize User Input used in Document Manipulation Expression
- ASCQM Sanitize User Input used in Document Navigation Expression
- ASCQM Sanitize User Input used in Expression Language Statement
- ASCQM Sanitize User Input used in Loop Condition
- ASCQM Sanitize User Input used in Path Manipulation
- ASCQM Sanitize User Input used in SQL Access
- ASCQM Sanitize User Input used in System Command
- ASCQM Sanitize User Input used in User Output
- ASCQM Sanitize User Input used to access Directory Resources

7.7.2 Number of distinct direct callers

The number of distinct direct callers shall be calculated as follows:

1. identify a code element.
2. build the set of code elements calling it.
3. compute the size of the set.

7.7.3 Measure specifications

1) The set of direct callers of any code element shall be determined as follows.

- the applicable call links shall be identified by a first **smm:OCLOperation <measureElement xmi:type="smm:OCLOperation" xmi:id="CallingAction" name="CallingAction" trait="ExposureSizing" shortDescription="" body="((oclIsTypeOf(kdm:action::CallableRelations) or**

oclIsTypeOf(kdm:action::DataRelations)) and to = self)" context="kdm:code::AbstractCodeElement"/>

- the callers shall be identified by a second **smm:OCLOperation <measureElement xmi:type="smm:OCLOperation" xmi:id="CallingCodeElements" name="CallingCodeElements" trait="ExposureSizing" shortDescription="" body="(self.CallingAction.from())" context="kdm:code::AbstractCodeElement"/>**

2) The number of distinct direct callers of any code element shall be determined as follows.

- the size of the set of callers shall be computed by an **smm:Operation <measureElement xmi:type="smm:OCLOperation" xmi:id="CallingCodeElementsNumber" name="CallingCodeElementsNumber" trait="ExposureSizing" shortDescription="" body="CallingCodeElements()->size()" context="kdm:code::AbstractCodeElement"/>**

3) To measure the number of distinct callers for all implementation roles, the following measures shall apply the specified **smm:Operation** to the identified exposed role; e.g., with **ASCQM Check Index of Array Access VariableDeclarationStatement Code Elements**

- an **smm:DirectMeasure** uses the **smm:OCLOperation** on the **smm:Scope <measureElement xmi:type="smm:OCLOperation" xmi:id="id.sfgd.34.VariableDeclarationStatement_direct_exposure" name="ASCQM Check Index of Array Access VariableDeclarationStatement Direct Exposure" unit="Integer" scope="id.sfgd.34.VariableDeclarationStatement_code_elements" trait="ExposureSizing" category="id.cat.277 id.cat.278" shortDescription="Number of direct callers to ASCQM Check Index of Array Access VariableDeclarationStatement Direct Exposure" operation="CallingCodeElementsNumber"/>**

A **smm:DirectMeasure** measure (whose key is the Detection Pattern with a '*_direct_exposure*' suffix) shall be defined for each pattern role from *ASCQM standard*.

7.7.4 Number of distinct call paths

The number of distinct call paths shall be computed similar to the McCabe Cyclomatic Complexity formula (CC = E – N + p) as follows.

1. identify a code element,
2. identify the call paths towards the code element,
3. compute the number of nodes (N),
4. compute the number of entry nodes to compute the number of edges (E) needed to cycle back to the starting code element in order that the number of components is 1,
5. compute the number of edges (E),
6. subtract the number of nodes (N) from the sum of the number of edges (E) and the number of entry nodes,
7. add 1 to the difference to get the number of distinct call paths

7.7.5 Measure specifications

A call graph for selected code elements shall be developed using the **:OCLOperation** from the previous paragraph.

- the call graph as recursive callers, identified by a first **smm:OCLOperation <measureElement xmi:type="smm:OCLOperation" xmi:id="CallingGraph" name="CallingGraph" trait="ExposureSizing" shortDescription="" body="(closure(CallingCodeElements()))" context="kdm:code::AbstractCodeElement"/>**

The number of distinct call paths of any code element shall be computed as:

- the number of nodes, computed by a **smm:DirectMeasure**
  <measureElement xmi:type="smm:DirectMeasure" xmi:id="CallingGraphNodeNumber" name="CallingGraphNodeNumber" trait="ExposureSizing" shortDescription="" operation="CallingGraphNodeNumber_value" baseMeasure1From="CallingGraphBranchingFactor_to_CallingGraphNodeNumber"/>
- and its **smm:Operation**
  <measureElement xmi:type="smm:Operation" xmi:id="CallingGraphNodeNumber_value" name="CallingGraphNodeNumber_value" trait="ExposureSizing" shortDescription="" body="CallingGraph()->select(e: kdm:code::AbstractCodeElement)->size()" language="OCL"/>
- the number of entry nodes, computed by a **smm:DirectMeasure**
  <measureElement xmi:type="smm:DirectMeasure" xmi:id="CallingGraphEntryNodeNumber" name="CallingGraphEntryNodeNumber" trait="ExposureSizing" shortDescription="" operation="CallingGraphEntryNodeNumber_value" baseMeasure2From="CallingGraphEdgeAndEntryNodeNumber_to_CallingGraphEdgeNumber"/>
- and its **smm:Operation**
  <measureElement xmi:type="smm:Operation" xmi:id="CallingGraphEntryNodeNumber_value" name="CallingGraphEntryNodeNumber_value" trait="ExposureSizing" shortDescription="" body="CallingGraph()->select(e: kdm:code::AbstractCodeElement | e.CallingCodeElementsNumber = 0)->size()" language="OCL"/>
- the number of edges, computed by a **smm:DirectMeasure**
  <measureElement xmi:type="smm:DirectMeasure" xmi:id="CallingGraphEdgeNumber" name="CallingGraphEdgeNumber" trait="ExposureSizing" shortDescription="" operation="CallingGraphEdgeNumber_value" baseMeasure1From="CallingGraphEdgeAndEntryNodeNumber_to_CallingGraphEdgeNumber"/>
- and its **smm:Operation**
  <measureElement xmi:type="smm:Operation" xmi:id="CallingGraphEdgeNumber_value" name="CallingGraphEdgeNumber_value" trait="ExposureSizing" shortDescription="" body="CallingGraph()->select(e1, e2: kdm:code::AbstractCodeElement | e1.CallingAction()->includes(e2))->size()" language="OCL"/>
- the sum of the number of edges and the number of entry nodes, computed by a first **smm:BinaryMeasure**
  <measureElement xmi:type="smm:BinaryMeasure" xmi:id="CallingGraphEdgeAndEntryNodeNumber" name="CallingGraphEdgeAndEntryNodeNumber" functor="plus" unit="Integer" scope="kdm:code::AbstractCodeElement" trait="ExposureSizing" shortDescription="Calling graph number of edges and entry nodes" baseMeasure1To="CallingGraphEdgeAndEntryNodeNumber_to_CallingGraphEdgeNumber" baseMeasure2To="CallingGraphEdgeAndEntryNodeNumber_to_CallingGraphEdgeNumber"

baseMeasure2From="CallingGraphBranchingFactor_to_CallingGraphEdgeAndEntryNodeNumber"/>

- the difference of the number of nodes from edges and entry nodes, computed by a second **smm:BinaryMeasure**
  <measureElement xmi:type="smm:BinaryMeasure" xmi:id="CallingGraphBranchingFactor" name="CallingGraphBranchingFactor" functor="minus" unit="Integer" scope="kdm:code::AbstractCodeElement" trait="ExposureSizing" shortDescription="Calling graph branching factor" baseMeasure1To="CallingGraphBranchingFactor_to_CallingGraphNodeNumber" baseMeasure2To="CallingGraphBranchingFactor_to_CallingGraphEdgeAndEntryNodeNumber" rescaleTo="CallingGraphBranchingFactor_to_GraphCallPathNumber"/>
- the number of distinct call paths, computed by an **smm:RescaledMeasure**
  <measureElement xmi:type="smm:RescaledMeasure" xmi:id="GraphCallPathNumber" name="GraphCallPathNumber" unit="Integer" scope="kdm:code::AbstractCodeElement" trait="ExposureSizing" shortDescription="Number of call paths to the Code Element" offset="1" multiplier="1" rescaleFrom="CallingGraphBranchingFactor_to_GraphCallPathNumber" rescaleTo="GraphCallPathNumber_to_LogGraphCallPathNumber"/>
- the logarithmic transformation of the number of distinct call paths, computed by an **smm:RescaledMeasure**
  <measureElement xmi:type="smm:RescaledMeasure" xmi:id="LogGraphCallPathNumber" name="LogGraphCallPathNumber" unit="Real" scope="kdm:code::AbstractCodeElement" trait="ExposureSizing" shortDescription="Log of the number of call paths to the Code Element" operation="log(GraphCallPathNumber)" rescaleFrom="GraphCallPathNumber_to_LogGraphCallPathNumber" rescaleTo="LogGraphCallPathNumber_to_id.sfgd.100.Table_exposure …" />

Finally, to measure the Exposure for all Detection Pattern Occurrences, the following measures shall apply the specified **:RescaleMeasure** to the identified role.

E.g., with **ASCQM Check Index of Array Access VariableDeclarationStatement Code Elements**

- an **smm:RescaledMeasure** uses the **smm:RescaledMeasure** on the **smm:Scope**
  <measureElement xmi:type="smm:RescaledMeasure" xmi:id="id.sfgd.34.VariableDeclarationStatement_exposure" name="ASCQM Check Index of Array Access VariableDeclarationStatement Exposure" unit="Real" scope="id.sfgd.34.VariableDeclarationStatement_code_elements" trait="ExposureSizing" category="id.cat.277 id.cat.278" shortDescription="Exposure of ASCQM Check Index of Array Access VariableDeclarationStatement Exposure (measured as 1 plus the log of the number of call paths to them)" offset="1" multiplier="1" baseMeasureFrom="id.sfgd.34.exposure_overhead_average_to_id.sfgd.34.VariableDeclarationStatement_exposure" rescaleFrom="LogGraphCallPathNumber_to_id.sfgd.34.VariableDeclarationStatement_exposure"/>

7.8 Concentration and Sharing Opportunity

7.8.1 Overview of Concentration

The Concentration shall be computed as follows:

Count the number of Occurrences of each of the Detection Pattern role.

E.g., with **ASCQM Check Index of Array Access role VariableDeclarationStatement**

- defined by an **smm:DirectMeasure**
  <measureElement xmi:type="smm:DirectMeasure" xmi:id="id.sfgd.34.VariableDeclarationStatement_concentration" name="ASCQM Check Index of Array Access VariableDeclarationStatement Concentration" unit="Integer" scope="id.sfgd.34.VariableDeclarationStatement_code_elements" trait="SharingLevelEstimating" category="id.cat.277 id.cat.278" shortDescription=" ASCQM Check Index of Array Access VariableDeclarationStatement Concentration (measured as the number of occurrences they are involved in)" operation="NumberOfOccurrences" rescaleTo="id.sfgd.34.VariableDeclarationStatement_concentration_to_id.sfgd.34.VariableDeclarationStatement_sharing"/>
- relying on an **smm:Operation**
  <measureElement xmi:type="smm:Operation" xmi:id="NumberOfOccurrences" name="NumberOfOccurrences" body="self.A_Binding_fulfilledBy::Binding()->select(b:Binding | p.A_PatternInstance_fulfillments::PatternInstance.instanceOf.isInASCQM)->size()" language="OCL"/>
- which uses the following **smm:OCLOperation**
  <measureElement xmi:type="smm:OCLOperation" xmi:id="isInASCQM" name="isInASCMM" body="Set{
  'ascqm:id.sfgd.34','ascqm:id.sfgd.25','ascqm:id.sfgd.19','ascqm:id.sfgd.26','ascqm:id.sfgd.15','ascqm:id.sfgd.24','ascqm:id.sfgd.27','ascqm:id.sfgd.29','ascqm:id.sfgd.30','ascqm:id.sfgd.12','ascqm:id.sfgd.79','ascqm:id.sfgd.338','ascqm:id.sfgd.122','ascqm:id.sfgd.141','ascqm:id.sfgd.38','ascqm:id.sfgd.120','ascqm:id.sfgd.321','ascqm:id.sfgd.41','ascqm:id.sfgd.327','ascqm:id.sfgd.340','ascqm:id.sfgd.44','ascqm:id.sfgd.45','ascqm:id.sfgd.57','ascqm:id.sfgd.59','ascqm:id.sfgd.60','ascqm:id.sfgd.232','ascqm:id.sfgd.341','ascqm:id.sfgd.61','ascqm:id.sfgd.69','ascqm:id.sfgd.78','ascqm:id.sfgd.138','ascqm:id.sfgd.157','ascqm:id.sfgd.260','ascqm:id.sfgd.344','ascqm:id.sfgd.127','ascqm:id.sfgd.128','ascqm:id.sfgd.106','ascqm:id.sfgd.125','ascqm:id.sfgd.109','ascqm:id.sfgd.154','ascqm:id.sfgd.121','ascqm:id.sfgd.114','ascqm:id.sfgd.123','ascqm:id.sfgd.140','ascqm:id.sfgd.81','ascqm:id.sfgd.92','ascqm:id.sfgd.82','ascqm:id.sfgd.85','ascqm:id.sfgd.133','ascqm:id.sfgd.261','ascqm:id.sfgd.238','ascqm:id.sfgd.148','ascqm:id.sfgd.149','ascqm:id.sfgd.107','ascqm:id.sfgd.126','ascqm:id.sfgd.83','ascqm:id.sfgd.189','ascqm:id.sfgd.328','ascqm:id.sfgd.329','ascqm:id.sfgd.333','ascqm:id.sfgd.334','ascqm:id.sfgd.110','ascqm:id.sfgd.335','ascqm:id.sfgd.136','ascqm:id.sfgd.290','ascqm:id.sfgd.134','ascqm:id.sfgd.297','ascqm:id.sfgd.301','ascqm:id.sfgd.326','ascqm:id.sfgd.190','ascqm:id.sfgd.305','ascqm:id.sfgd.313','ascqm:id.sfgd.312','ascqm:id.sfgd.317','ascqm:id.sfgd.337','ascqm:id.sfgd.320','ascqm:id.sfgd.142','ascqm:id.sfgd.339','ascqm:id.sfgd.72','ascqm:id.sfgd.319','ascqm:id.sfgd.318','ascqm:id.sfgd.332','ascqm:id.sfgd.322','ascqm:id.sfgd.323','ascqm:id.sfgd.93','ascqm:id.sfgd.96','ascqm:id.sfgd.145','ascqm:id.sfgd.146','ascqm:id.sfgd.95','ascqm:id.sfgd.150','ascqm:id.sfgd.151','ascqm:id.sfgd.94','ascqm:id.sfgd.119','ascqm:id.sfgd.143','ascqm:id.sfgd.252','ascqm:id.sfgd.144','ascqm:id.sfgd.147','ascqm:id.sfgd.152','ascqm:id.sfgd.331','ascqm:id.sfgd.330','ascqm:id.sfgd.343','ascqm:id.sfgd.97','ascqm:id.sfgd.100','ascqm:id.sfgd.99','ascqm:id.sfgd.101','ascqm:id.sfgd.105','ascqm:id.sfgd.108','ascqm:id.sfgd.111','ascqm:id.sfgd.117','ascqm:id.sfgd.118','ascqm:id.sfgd.139','ascqm:id.sfgd.91','ascqm:id.sfgd.129','ascqm:id.sfgd.113','ascqm:id.sfgd.159','ascqm:id.sfgd.153','ascqm:id.sfgd.156','ascqm:id.sfgd.102','ascqm:id.sfgd.98','ascqm:id.sfgd.112','ascqm:id.sfgd.116','ascqm:id.sfgd.130','ascqm:id.sfgd.124','ascqm:id.sfgd.132','ascqm:id.sfgd.131','ascqm:id.sfgd.87','a

scqm:id.sfgd.88','ascqm:id.sfgd.90','ascqm:id.sfgd.84','ascqm:id.sfgd.155','ascqm:id.sfgd.336','ascqm:id.sfgd.137','ascqm:id.sfgd.103','ascqm:id.sfgd.104','ascqm:id.sfgd.135'}->includes(self.id)" context="spms:Definitions::PatternDefinition"/>

7.8.2 Sharing Opportunities

The Sharing Opportunity shall be computed as follows:

The inverse of the Concentration of each of the Detection Pattern role.

E.g., with **ASCQM Check Index of Array Access role VariableDeclarationStatement**

- an **smm:RescaledMeasure**
  <measureElement xmi:type="smm:RescaledMeasure" xmi:id="id.sfgd.34.VariableDeclarationStatement_sharing" name="ASCQM Check Index of Array Access VariableDeclarationStatement Sharing Opportunity (measured as the inverse of the number of occurrences the code elements supporting the role are involved in)" unit="Real" scope="id.sfgd.34.VariableDeclarationStatement_code_elements" operation="1 / id.sfgd.34.VariableDeclarationStatement_concentration" offset="" multiplier="" baseMeasureFrom="id.sfgd.34.sharing_opportunity_average_to_id.sfgd.34.VariableDeclarationStatement_sharing" rescaleFrom="id.sfgd.34.VariableDeclarationStatement_concentration_to_id.sfgd.34.VariableDeclarationStatement_sharing"/>

7.8.3 Measure specifications

For each implementation role from *ASCQM* standard Detection Patterns, an **smm:OCLOperation** (whose key is the Detection Pattern key with a '*_concentration*' suffix) and an **smm:RescaledMeasure** (whose key is the Detection Pattern key with a '*_sharing*' suffix) shall be defined.

For each implementation role, the **smm:Scope** (named as the role name with a '*_code_elements*' suffix), and its recognizer **smm:Operation** (named as the role name with a '*_code_elements_recognizer'* suffix) will be reused in the current process.

7.9 Occurrence Gap Size

7.9.1 Definition of Occurrence Gap Size

This sub-clause shall only be applicable when the Detection Pattern relies on roles that compare existing values to threshold values that are not to be exceeded. The Occurrence Gap Size is the extent of the gap to be closed to remediate the Detection Pattern Occurrence, measured as the difference between the existing value and the threshold value.

The affected Detection Patterns are:

- ASCQM Ban Excessive Complexity of Data Resource Access
- ASCQM Ban Excessive Number of Children
- ASCQM Ban Excessive Number of Concrete Implementations to Inherit From
- ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code
- ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure
- ASCQM Ban Excessive Number of Index on Columns of Large Tables
- ASCQM Ban Excessive Number of Inheritance Levels
- ASCQM Ban Excessive Size of Index on Columns of Large Tables

- ASCQM Limit Number of Aggregated Non-Primitive Data Types
- ASCQM Limit Number of Data Access
- ASCQM Limit Number of Outward Calls
- ASCQM Limit Number of Parameters
- ASCQM Limit Size of Operations Code
- ASCQM Limit Volume of Commented-Out Code
- ASCQM Limit Volume of Similar Code

For each Occurrence of these Detection Patterns, the Occurrence Gap Size shall be computed as follows:

1. Retrieve the value of the roles modeling the exceeding values
2. Retrieve the value of the roles modeling the threshold values
3. Compute the difference.

The difference formulae are:

- ASCQM Ban Excessive Complexity of Data Resource Access Gap Size :
  (id.sfgd.105.NumberOfTables – id.sfgd.105.MaxNumberOfTables) +
  (id.sfgd.105.NumberOfSubqueries - id.sfgd.105.MaxNumberOfSubqueries)

- ASCQM Ban Excessive Number of Children Gap Size :
  id.sfgd.112.NumberOfChildren - id.sfgd.112.MaxNumberOfChildren

- ASCQM Ban Excessive Number of Concrete Implementations to Inherit From Gap Size :
  id.sfgd.88.NumberOfConcreteClasseInheritances - id.sfgd.88.MaxNumberOfConcreteClass~

- ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code Gap Size :
  id.sfgd.118.NumberOfDataAccess - id.sfgd.118.MaxNumberOfDataAccess

- ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure Gap Size : id.sfgd.117.NumberOfDataAccess - id.sfgd.117.MaxNumberOfDataAccess

- ASCQM Ban Excessive Number of Index on Columns of Large Tables Gap Size :
  id.sfgd.101.NumberOfIndexes - id.sfgd.101.MaxNumberOfIndexes

- ASCQM Ban Excessive Number of Inheritance Levels Gap Size :
  id.sfgd.116.NumberOfInheritanceLevels - id.sfgd.116.MaxNumberOfInheritanceLevels

- ASCQM Ban Excessive Size of Index on Columns of Large Tables Gap Size :
  id.sfgd.100.TotalSizeOfIndexes - id.sfgd.100.MaxTotalSizeOfIndexes

- ASCQM Limit Number of Aggregated Non-Primitive Data Types Gap Size :
  id.sfgd.111.NumberOfNonPrimitiveMembers -
  id.sfgd.111.MaxNumberOfNonPrimitiveMembers

- ASCQM Limit Number of Data Access Gap Size :
  id.sfgd.98.NumberOfDataAccess - id.sfgd.98.MaxNumberOfDataAccess

- ASCQM Limit Number of Outward Calls Gap Size :
  id.sfgd.90.NumberOfOutwardCalls - id.sfgd.90.MaxNumberOfOutwardCalls

- ASCQM Limit Number of Parameters Gap Size :
  id.sfgd.131.NumberOfParameter - id.sfgd.131.MaxNumberOfParameter

- ASCQM Limit Size of Operations Code Gap Size :
  id.sfgd.153.NumberOfNonEmptyLinesOfCode -
  id.sfgd.153.MaxNumberOfNonEmptyLinesOfCode

- ASCQM Limit Volume of Commented-Out Code Gap Size : id.sfgd.159.PercentageOfCommentedOutCode - id.sfgd.159.MaxPercentageOfCommentedOutC~
- ASCQM Limit Volume of Similar Code Gap Size : id.sfgd.156.PercentageOfSimilarElements - id.sfgd.156.MaxPercentageOfSimilarElements

They require to get values from the following roles:

- ASCQM Ban Excessive Complexity of Data Resource Access Gap Size 1 id.sfgd.105.NumberOfTables
- ASCQM Ban Excessive Complexity of Data Resource Access Gap Size 1 id.sfgd.105.MaxNumberOfTables
- ASCQM Ban Excessive Complexity of Data Resource Access Gap Size 2 id.sfgd.105.NumberOfSubqueries
- ASCQM Ban Excessive Complexity of Data Resource Access Gap Size 2 id.sfgd.105.MaxNumberOfSubqueries
- ASCQM Ban Excessive Number of Children Gap Size id.sfgd.112.NumberOfChildren
- ASCQM Ban Excessive Number of Children Gap Size id.sfgd.112.MaxNumberOfChildren
- ASCQM Ban Excessive Number of Concrete Implementations to Inherit From Gap Size id.sfgd.88.NumberOfConcreteClasseInheritances
- ASCQM Ban Excessive Number of Concrete Implementations to Inherit From Gap Size id.sfgd.88.MaxNumberOfConcreteClasseInheritances
- ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code Gap Size id.sfgd.118.NumberOfDataAccess
- ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code Gap Size id.sfgd.118.MaxNumberOfDataAccess
- ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure Gap Size id.sfgd.117.NumberOfDataAccess
- ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure Gap Size id.sfgd.117.MaxNumberOfDataAccess
- ASCQM Ban Excessive Number of Index on Columns of Large Tables Gap Size id.sfgd.101.NumberOfIndexes
- ASCQM Ban Excessive Number of Index on Columns of Large Tables Gap Size id.sfgd.101.MaxNumberOfIndexes
- ASCQM Ban Excessive Number of Inheritance Levels Gap Size id.sfgd.116.NumberOfInheritanceLevels
- ASCQM Ban Excessive Number of Inheritance Levels Gap Size id.sfgd.116.MaxNumberOfInheritanceLevels
- ASCQM Ban Excessive Size of Index on Columns of Large Tables Gap Size id.sfgd.100.TotalSizeOfIndexes

- ASCQM Ban Excessive Size of Index on Columns of Large Tables Gap Size id.sfgd.100.MaxTotalSizeOfIndexes
- ASCQM Limit Number of Aggregated Non-Primitive Data Types Gap Size id.sfgd.111.NumberOfNonPrimitiveMembers
- ASCQM Limit Number of Aggregated Non-Primitive Data Types Gap Size id.sfgd.111.MaxNumberOfNonPrimitiveMembers
- ASCQM Limit Number of Data Access Gap Size id.sfgd.98.NumberOfDataAccess
- ASCQM Limit Number of Data Access Gap Size id.sfgd.98.MaxNumberOfDataAccess
- ASCQM Limit Number of Outward Calls Gap Size id.sfgd.90.NumberOfOutwardCalls
- ASCQM Limit Number of Outward Calls Gap Size id.sfgd.90.MaxNumberOfOutwardCalls
- ASCQM Limit Number of Parameters Gap Size id.sfgd.131.NumberOfParameter
- ASCQM Limit Number of Parameters Gap Size id.sfgd.131.MaxNumberOfParameter
- ASCQM Limit Size of Operations Code Gap Size id.sfgd.153.NumberOfNonEmptyLinesOfCode
- ASCQM Limit Size of Operations Code Gap Size id.sfgd.153.MaxNumberOfNonEmptyLinesOfCode
- ASCQM Limit Volume of Commented-Out Code Gap Size id.sfgd.159.PercentageOfCommentedOutCode
- ASCQM Limit Volume of Commented-Out Code Gap Size id.sfgd.159.MaxPercentageOfCommentedOutCode
- ASCQM Limit Volume of Similar Code Gap Size id.sfgd.156.PercentageOfSimilarElements
- ASCQM Limit Volume of Similar Code Gap Size id.sfgd.156.MaxPercentageOfSimilarElements

To do so, an **smm:Operation** and an **smm:DirectMeasure** shall be defined as follows (using an example with **ASCQM Ban Excessive Size of Index on Columns of Large Tables role TotalSizeOfIndexes**):

- **<measureElement xmi:type="smm:DirectMeasure" xmi:id="id.sfgd.100.TotalSizeOfIndexes" name="ASCQM Ban Excessive Size of Index on Columns of Large Tables MaxTotalSizeOfIndexes Measure" unit="Integer" scope="id.sfgd.100.scope" trait="OccurrenceGapSizing" category="id.cat.279" operation="id.sfgd.100.TotalSizeOfIndexes_value" />**
- relying on **<measureElement xmi:type="smm:Operation" xmi:id="id.sfgd.100.TotalSizeOfIndexes_value" name="ASCQM Ban Excessive Size of Index on Columns of Large Tables MaxTotalSizeOfIndexes Operation to retrieve the value" scope="id.sfgd.100.scope" trait="OccurrenceGapSizing" category="id.cat.279" body="ascqm:id.sfgd.100.TotalSizeOfIndexes.A_boundTo_Binding::Binding().fulfilledBy()" language="OCL"/>**

The Occurrence Gap Size is then an **smm:BinaryMeasure** computing the difference according to the formulae above using an example with ASCMM-CWE-1121:

- **<measureElement xmi:type="smm:BinaryMeasure" xmi:id="id.sfgd.100_1.gap_size" name="ASCQM Ban Excessive Size of Index on Columns of Large Tables Gap Size " functor="minus" unit="Integer" scope="id.sfgd.100.scope" trait="OccurrenceGapSizing" category="id.cat.279" shortDescription="Occurrence gap size for ASCQM Ban Excessive Size of Index on Columns of Large Tables regarding the TotalSizeOfIndexes " baseMeasure1To="id.sfgd.100_1.gap_size_to_id.sfgd.100.TotalSizeOfIndexes" baseMeasure2To="id.sfgd.100_1.gap_size_to_id.sfgd.100.MaxTotalSizeOfIndexes" baseMeasureFrom="id.sfgd.100.adjustment_factor_to_id.sfgd.100_1.gap_size"/>**

7.9.2 Measure specifications

For each applicable patterns from the *ASCQM* standard patterns (listed above), an **smm:BinaryMeasure** (whose key is the Detection Pattern with a '_x.gap_size' suffix) shall be defined, with 'x' being an integer index used to handle multiple gaps for a single pattern (in current version of *ASCQM* standard, only ASCQM Ban Excessive Complexity of Data Resource Access Detection Pattern features two gaps).

For each applicable implementation role (listed above), the **smm:DirectMeasure** (named as the role name without any suffix), and its **smm:Operation** (named as the role name with a '_value' suffix) shall be defined.

7.10 Evolution

This sub-clause shall only be applicable when two revisions of the software are available for measurement.

7.10.1 Involved Code Elements

The Evolution of involved code elements shall be computed as follows:

1. For each implementation role, use the defined scope to identify code elements.
2. For each code element, its status shall be identified as added, updated, deleted, or unchanged based on the following guidelines.
    - 'added' in latest Revision when there is no code element which evolved into it from a previous Revision.
    - 'deleted' from a previous Revision when there is no code element in the latest Revision into which it evolved.
    - 'updated' in latest Revision where the evidence in the source code that its implementation evolved from its instantiation in a previous release.
    - 'unchanged' if the code element remains identical in the two revisions.

To identify the Evolution of any code element, a set of **smm:OCLOperation** for each code element shall be determined.

- added **<measureElement xmi:type="smm:OCLOperation" xmi:id="isAddedElement" name="isAddedElement" trait="EvolutionStatus" shortDescription="Evolutions status measured code element: TRUE if added between revisions" context="kdm:Core::Element" body="(isInLatestRevision and not fromRevisionMeasurementScope()->exists(e:kdm:Core::Element | e.evolvedTo = self))"/>**
- deleted **<measureElement xmi:type="smm:OCLOperation" xmi:id="isDeletedElement" name="isDeletedElement" trait="EvolutionStatus" shortDescription="Evolutions status**

measured code element: TRUE if deleted between revisions" context="kdm:Core::Element" body="(isInPreviousRevision and not toRevisionMeasurementScope()->exists(e:kdm:Core::Element | e.evolvedFrom = self))"/>

- updated
  <measureElement xmi:type="smm:OCLOperation" xmi:id="isUpdatedElement" name="isUpdatedElement" trait="EvolutionStatus" shortDescription="Evolutions status measured code element: TRUE if updated between revisions" context="kdm:Core::Element" body="(isInLatestRevision and toRevisionMeasurementScope()->exists(e:kdm:Core::Element | e.evolvedTo = self and self.source != e.source))"/>

- unchanged
  <measureElement xmi:type="smm:OCLOperation" xmi:id="isUnchangedElement" name="isUnchangedElement" trait="EvolutionStatus" shortDescription="Evolutions status measured code element: TRUE if unchanged between revisions" context="kdm:Core::Element" body="(isInLatestRevision and not (isUpdatedElement or isAddedElement))"/>

7.10.2 Occurrence Gap Size

The computation of the Evolution of each Detection Pattern Occurrence shall include the following additional steps.

1. The analyzer shall check to determine if the roles are implemented by code elements evolved from code elements implementing the same roles in the previous release.
   - either with unchanged code elements, identified via a first **smm:OCLOperation <measureElement xmi:type="smm:OCLOperation" xmi:id="hasAllItsCodeElementsUnchangedFromCodeElementsInBindingOfSameRole" name="hasAllItsCodeElementsEvolvedFromCodeElementsInBindingOfSameRole" trait="EvolutionStatus" shortDescription="Evolutions status role implementation: TRUE if all code elements unchanged between revisions and implementing a binding of the same role in previous release" context="spms:Observations::Binding" body="self.fullfiled()->forAll(e: kdm:Core::Element | e.isUnchangedElement and e.evolvedFrom.A_Binding_fulfilledBy::Binding()->exist(b: Binding | b.boundTo = self.boundTo))"/>**
   - either with unchanged or updated code elements, identified via a second **smm:OCLOperation <measureElement xmi:type="smm:OCLOperation" xmi:id="hasAllItsCodeElementsEvolvedFromCodeElementsInBindingOfSameRole" name="hasAllItsCodeElementsEvolvedFromCodeElementsInBindingOfSameRole" trait="EvolutionStatus" shortDescription="Evolutions status role implementation: TRUE if all code elements implementing a binding of the same role in previous release" context="spms:Observations::Binding" body="self.fullfiled()->forAll(e: kdm:Core::Element | e.evolvedFrom.A_Binding_fulfilledBy::Binding()->exist(b: Binding | b.boundTo = self.boundTo))"/>**
2. An Occurrence shall be considered as:
   - unchanged, if all its roles are implemented by unchanged code elements evolved from code elements implementing the same roles in the previous release, identified via a first **smm:OCLOperation**

<measureElement xmi:type="smm:OCLOperation" xmi:id="isUnchangedOccurrence" name="isUnchangedOccurrence" trait="EvolutionStatus" shortDescription="Evolutions status occurrence: TRUE if unchanged between revisions" context="spms:Observations::PatternInstance" body="self.fulfillments()->forAll(b: spms:Observations::Binding | b.hasAllItsCodeElementsUnchangedFromCodeElementsInBindingOfSameRole)"/>

- updated, if not unchanged and all its roles are implemented by code elements evolved from code elements implementing the same roles in the previous release, identified via a second **smm:OCLOperation**
  <measureElement xmi:type="smm:OCLOperation" xmi:id="isUpdatedOccurrence" name="isUpdatedOccurrence" trait="EvolutionStatus" shortDescription="Evolutions status occurrence: TRUE if updated between revisions" context="spms:Observations::PatternInstance" body="self.fulfillments()->forAll(b: spms:Observations::Binding | b.hasAllItsCodeElementsUnchangedFromCodeElementsInBindingOfSameRole) and not self.isUnchangedOccurrence"/>
- added, if in the "ToRevision" revision but not updated nor unchanged, identified via a third **smm:OCLOperation**
  <measureElement xmi:type="smm:OCLOperation" xmi:id="isAddedOccurrence" name="isAddedOccurrence" trait="EvolutionStatus" shortDescription="Evolutions status occurrence: TRUE if added between revisions" context="spms:Observations::PatternInstance" body="self.isInLatest and not self.isUnchangedOccurrence and not self.isUpdatedOccurrence"/>

7.11 Adjustment Factor

For each Detection Pattern Occurrence, the Adjustment Factor shall be calculated as the simple product of the following contributions:

- Complexity overhead average, across all implementation roles,
- Technological Diversity,
- Sharing opportunity average, across all implementation roles
- Occurrence Gap Size, when applicable

Note that the Evolution and Exposure information is not used for adjustments in ATDM, but can be used in CTDM.

7.11.1 Complexity Overhead Average Contribution

The contribution from the complexity overhead specified in Sub-clause 7.6 for each implementation role is a simple average.

E.g., with *ASCQM Check Index of Array Access*:

<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.sfgd.34.complexity_overhead_average" name="Occurrence Complexity Overhead Average of ASCQM Check Index of Array Access" unit="Real" scope="id.sfgd.34.scope" trait="ComplexityEstimating" category="id.cat.277 id.cat.278" shortDescription="Complexity overhead average of an occurrence of ASCQM Check Index of Array Access (measured as the AEP complexity overhead when compared to simplest complexity)" accumulator="average" baseMeasureTo="id.sfgd.34.complexity_overhead_average_to_id.sfgd.34.ArrayAccessStatement_complexity_overhead

id.sfgd.34.complexity_overhead_average_to_id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_complexity_overhead
id.sfgd.34.complexity_overhead_average_to_id.sfgd.34.VariableDeclarationStatement_complexity_overhead"
baseMeasureFrom="id.sfgd.34.adjustment_factor_to_id.sfgd.34.complexity_overhead_average"/>

7.11.2 Measure specifications

An **smm:CollectiveMeasure** measure (whose key is the Detection Pattern with a '*.complexity_overhead_average*' suffix) shall be defined for each source code pattern from *ASCQM standard*, as illustrated with the *ASCQM Check Index of Array Access* pattern above.

7.11.3 Exposure Overhead Average Contribution

The contribution from the Exposure or Direct Exposure specified in Sub-clause 7.7 for each implementation role is a simple average. It is considered an overhead vis-à-vis the 'best case scenario' in 'well-implemented' code where the Exposure value is "1".

- E.g. with *ASCQM Check Index of Array Access*:direct exposure **<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.sfgd.34.direct_exposure_overhead_average" name="Occurrence Direct Exposure Overhead Average of ASCQM Check Index of Array Access" unit="Real" scope="id.sfgd.34.scope" trait="ExposureEstimating" category="id.cat.277 id.cat.278" shortDescription="Occurrence Direct Exposure Overhead Average of ASCQM Check Index of Array Access (measured as the direct exposure overhead when compared to simplest direct exposure of 1)" accumulator="average" baseMeasureTo="id.sfgd.34.direct_exposure_overhead_average_to_id.sfgd.34.ArrayAccessStatement_direct_exposure id.sfgd.34.direct_exposure_overhead_average_to_id.sfgd.34.VariableDeclarationStatement_direct_exposure"/>**
- **exposure <measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.sfgd.34.exposure_overhead_average" name="Occurrence Exposure Overhead Average of ASCQM Check Index of Array Access" unit="Real" scope="id.sfgd.34.scope" trait="ExposureEstimating" category="id.cat.277 id.cat.278" shortDescription="Exposure overhead average of an occurrence of ASCQM Check Index of Array Access (measured as the exposure overhead when compared to simplest exposure of 1)" accumulator="average" baseMeasureTo="id.sfgd.34.exposure_overhead_average_to_id.sfgd.34.ArrayAccessStatement_exposure id.sfgd.34.exposure_overhead_average_to_id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_exposure id.sfgd.34.exposure_overhead_average_to_id.sfgd.34.VariableDeclarationStatement_exposure" baseMeasureFrom="id.sfgd.34.adjustment_factor_to_id.sfgd.34.exposure_overhead_average"/>**

7.11.4 Measure Specifications

Two **smm:CollectiveMeasure** measure (whose key is the Detection Pattern with '.direct_exposure_overhead_average' and '.exposure_overhead_average' suffixes) shall be defined for each source code pattern from *ASCQM standard*, as illustrated with the *ASCQM Check Index of Array Access* pattern above.

7.11.5 Technological Diversity Contribution

The contribution from the Occurrence of Technological Diversity specified in Sub-clause 7.5 is direct, that is, the number of languages in which the Detection Pattern Occurrence is implemented is used as the Technological Diversity input to the Adjustment Factor calculation.

7.11.6 Sharing Opportunity Average Contribution

The contribution from the Sharing Opportunity specified in Sub-clause 7.8 for each implementation role is a simple average. It is considered an opportunity to share the Remediation Effort vis-à-vis the nominal situation where the Concentration value is 1.

E.g., with *ASCQM Check Index of Array Access*:

<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.sfgd.34.sharing_opportunity_average" name="Occurrence Sharing Opportunity Average of ASCQM Check Index of Array Access" unit="Real" scope="id.sfgd.34.scope" trait="SharingLevelEstimating" category="id.cat.277 id.cat.278" shortDescription="Sharing opportunity average of an occurrence of ASCQM Check Index of Array Access (measured as the distinct occurrences sharing code elements)" accumulator="average" baseMeasureTo="id.sfgd.34.sharing_opportunity_average_to_id.sfgd.34.ArrayAccessStatement_sharing id.sfgd.34.sharing_opportunity_average_to_id.sfgd.34.PathFromDeclarationStatementToUseAsAnIndexStatement_sharing id.sfgd.34.sharing_opportunity_average_to_id.sfgd.34.VariableDeclarationStatement_sharing" baseMeasureFrom="id.sfgd.34.adjustment_factor_to_id.sfgd.34.sharing_opportunity_average"/>

7.11.7 Measure specifications

A **smm:CollectiveMeasure** measure (whose key is the Detection Pattern with a '*.sharing_opportunity_average*' suffix) shall be defined for each Detection Pattern from *ASCQM standard*, as illustrated with the *ASCQM Check Index of Array Access* pattern above.

7.11.8 Occurrence Gap Size Contribution

The contribution from the Occurrence Gap Size specified in Sub-clause 7.9 is either:

- direct, that is, the difference between existing value and threshold value that has been exceeded is used as input to the Adjustment Factor calculation.
- Indirect, via a transformation (log base 2 in this version of the specifications), for a selection of four Detection Patterns, when the remediation of the issue is generally not done by reducing the gap of 1 point at a time. These four Detection Patterns are:
  - ASCQM Ban Excessive Size of Index on Columns of Large Tables,
  - ASCQM Limit Algorithmic Complexity via Cyclomatic Complexity Value, ASCQM Limit Size of Operations Code,
  - ASCQM Limit Volume of Similar Code, and
  - ASCQM Limit Volume of Commented-Out Code.

The transformation is performed by an **smm:RescaledMeasure**, which transform the result of the difference between exceeding value and threshold value.

E.g., ASCQM Ban Excessive Size of Index on Columns of Large Tables:

<measureElement xmi:type="smm:RescaledMeasure" xmi:id="id.sfgd.100_1.rescaled_gap_size" name="ASCQM Ban Excessive Size of Index on Columns of Large Tables Gap Size Rescaled (log2) " offset="" multiplier="" operation="log2(id.sfgd.100_1.gap_size)" unit="Real" scope="id.sfgd.100.scope" trait="OccurrenceGapSizing" category="id.cat.279" shortDescription="Rescaled occurrence gap size for ASCQM Ban Excessive Size of Index on Columns of Large Tables regarding the TotalSizeOfIndexes "/>

7.11.9 Adjustment Factor Computation

For each Detection Pattern Occurrence, the Adjustment Factor shall be computed as the product of all three or four contributions.

E.g., with *ASCQM Check Index of Array Access*:

<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.sfgd.34.adjustment_factor" name="Occurrence Adjustment Factor of ASCQM Check Index of Array Access" unit="Real" scope="id.sfgd.34.scope" trait="RemediationEffortEstimating" category="id.cat.277 id.cat.278" shortDescription="Contextual Factor to adjust Unadjusted Remediation Effort to remove one occurrence of ASCQM Check Index of Array Access" accumulator="product" isGapDependent="FALSE" baseMeasureTo="id.sfgd.34.adjustment_factor_to_id.sfgd.34.complexity_overhead_average id.sfgd.34.adjustment_factor_to_id.sfgd.34.sharing_opportunity_average id.sfgd.34.adjustment_factor_to_id.sfgd.34.technological_diversity" baseMeasure2From="id.sfgd.34.remediation_effort_to_id.sfgd.34.adjustment_factor"/>

E.g. with ASCQM Ban Excessive Number of Children, which features a gap size:

<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.sfgd.112.adjustment_factor" name="Occurrence Adjustment Factor of ASCQM Ban Excessive Number of Children" unit="Real" scope="id.sfgd.112.scope" trait="RemediationEffortEstimating" category="id.cat.280" shortDescription="Contextual Factor to adjust Unadjusted Remediation Effort to remove one occurrence of ASCQM Ban Excessive Number of Children" accumulator="product" isGapDependent="TRUE" baseMeasureTo="id.sfgd.112.adjustment_factor_to_id.sfgd.112.complexity_overhead_average id.sfgd.112.adjustment_factor_to_id.sfgd.112.sharing_opportunity_average id.sfgd.112.adjustment_factor_to_id.sfgd.112.technological_diversity id.sfgd.112.adjustment_factor_to_id.sfgd.112_1.gap_size" baseMeasure2From="id.sfgd.112.remediation_effort_to_id.sfgd.112.adjustment_factor"/>

E.g. with ASCQM Ban Excessive Size of Index on Columns of Large Tables, which features a rescaled gap size:

<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.sfgd.100.adjustment_factor" name="Occurrence Adjustment Factor of ASCQM Ban Excessive Size of Index on Columns of Large Tables" unit="Real" scope="id.sfgd.100.scope" trait="RemediationEffortEstimating" category="id.cat.279" shortDescription="Contextual Factor to adjust Unadjusted Remediation Effort to remove one occurrence of ASCQM Ban Excessive Size of Index on Columns of Large Tables" accumulator="product" isGapDependent="TRUE" baseMeasureTo="id.sfgd.100.adjustment_factor_to_id.sfgd.100.complexity_overhead_average id.sfgd.100.adjustment_factor_to_id.sfgd.100.sharing_opportunity_average

id.sfgd.100.adjustment_factor_to_id.sfgd.100.technological_diversity
id.sfgd.100.adjustment_factor_to_id.sfgd.100_1.rescaled_gap_size"
baseMeasure2From="id.sfgd.100.remediation_effort_to_id.sfgd.100.adjustment_factor"/>

Measure specifications

An **smm:CollectiveMeasure** measure (whose key is the Detection Pattern with an '*.adjustment_factor*' suffix) shall be defined for each Detection Pattern from the *ASCQM* standard, as illustrated with the *ASCQM Check Index of Array Access* pattern above.

7.12 Adjusted Remediation Effort

For each Occurrence, the adjusted Remediation Effort is simply the product of the Unadjusted Remediation Effort value from Sub-clause 31 by the Adjustment Factor value from Sub-clause 7.11. For example, with *ASCQM Check Index of Array Access*:

<measureElement xmi:type="smm:BinaryMeasure" xmi:id="id.sfgd.34.remediation_effort"
name="Occurrence Remediation Effort of ASCQM Check Index of Array Access" functor="multiply"
unit="effort(minutes)" scope="id.sfgd.34.scope" trait="RemediationEffortEstimating"
category="id.cat.277 id.cat.278" shortDescription="Effort to remove one occurrence of ASCQM Check Index of Array Access (in context)"
baseMeasure1To="id.sfgd.34.remediation_effort_to_id.sfgd.34.unadjusted_remediation_effort"
baseMeasure2To="id.sfgd.34.remediation_effort_to_id.sfgd.34.adjustment_factor"
baseMeasureFrom="id.sfgd.34.pattern_remediation_effort_to_id.sfgd.34.remediation_effort"/>

7.12.1 Measure specifications

An **smm:BinaryMeasure** measure (whose key is the Detection Pattern with a '*.remediation effort*' suffix) shall be defined for each source code pattern from *ASCQM standard*, as illustrated with the *ASCQM Check Index of Array Access* pattern above.

7.13 Quantification of Remediation Effort at the Detection Pattern level

The Pattern Remediation Effort values are simply the sum for each Detection Pattern of the Occurrence Remediation Effort values described in Sub-clause 7.13.

This summation shall be done with an **smm:CollectiveMeasure**. For example, with the *ASCQM Check Index of Array Access* pattern:

<measureElement xmi:type="smm:CollectiveMeasure"
xmi:id="id.sfgd.34.pattern_remediation_effort" name="Pattern Remediation Effort of ASCQM Check Index of Array Access" unit="effort(minutes)" scope="toRevisionMeasurementScope"
trait="RemediationEffortEstimating" category="id.cat.277 id.cat.278" shortDescription="Effort to remove all occurrences of ASCQM Check Index of Array Access (in context)" accumulator="sum"
baseMeasureTo="id.sfgd.34.pattern_remediation_effort_to_id.sfgd.34.remediation_effort"
baseMeasureFrom="id.atdm_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.wk.1.weakness_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.wk.2.weakness_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.wk.35.weakness_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.wk.4.weakness_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.wk.6.weakness_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.wk.8.weakness_remediation_effort_to_id.sfgd.34.pattern_remediation_effort"/>

Measure Specifications

An **smm:CollectiveMeasure** measure (whose key is the Detection Pattern with a '*_PatternRemediationEffort*' suffix) shall be defined for each source code pattern from *ASCQM standard*, as illustrated with the *ASCQM Check Index of Array Access* pattern above.

7.14 Quantification of Remediation Effort at the Weakness Level

7.14.1 Weakness Remediation Effort

Remediation Efforts shall be calculated for each of the ASCQM Weaknesses. The values shall be computed by summing the Remediation Efforts for applicable Detection Patterns associated to specific Weaknesses, as defined in the *ASCQM* standard.

E.g., for Weakness CWE-125 Out-of-bounds Read:

<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.wk.1.weakness_remediation_effort" name="Weakness CWE-125 Out-of-bounds Read Remediation Effort" unit="effort(minutes)" scope="toRevisionMeasurementScope" trait="RemediationEffortEstimating" category="id.cat.277 id.cat.278" shortDescription="Effort to remove all occurrences of CWE-125 Out-of-bounds Read weakness (measured as the sum of remediation efforts of all contributing detection patterns, directly or indirectly via child weaknesses)" accumulator="sum" baseMeasureTo="id.wk.1.weakness_remediation_effort_to_id.sfgd.34.pattern_remediation_effort "/>

7.14.2 Pattern Applicability Considerations

Although designed as technology-agnostic specifications, the *ASCQM* standard contains Detection Patterns that are not applicable to all programming languages. When a pattern is not applicable, there are no Detection Pattern Occurrences to process.

7.14.3 Shared Pattern Considerations

Detection Patterns are occasionally shared between Weaknesses (e.g., among Weaknesses within the same parent-child groups of Weaknesses). Each unique Detection Pattern Occurrence would only be fixed only once. Therefore, each Detection Pattern Occurrence must be counted only once.

E.g., ASCQM Ban Use of Expired Pointer Detection Pattern supports Weaknesses CWE-825 and CWE-119 (both in the parent-child group of Weakness CWE-119), as well as Weaknesses CWE-416 and CWE-672 (both in the parent-child group of Weakness CWE-672). Therefore, when computing Weakness CWE-119 Remediation Effort, ASCQM Ban Use of Expired Pointer Detection Pattern Remediation Effort shall be counted only once if this unique occurrence of the Detection Pattern triggers the identification of more than one Weakness.

7.15 Quantification of Remediation Effort for ASCQM Quality Characteristics

7.14.1 Quality Characteristic Remediation Effort

Remediation Efforts shall be calculated for each of the ASCQM Quality Characteristics.

- Reliability Remediation Effort Measure (RREM)
- Security Remediation Effort Measure (SREM)
- Performance Efficiency Remediation Effort Measure (PEREM)
- Maintainability Remediation Effort Measure (MREM)

The *MREM*, *RREM*, *PEREM*, and *SREM* values shall be computed by summing the Remediation Efforts for applicable Detection Patterns associated to Maintainability, Reliability, Performance Efficiency, and Security quality characteristics respectively, as defined in the *ASCQM* standard.

7.15.2 Pattern Applicability Considerations

Although designed as technology-agnostic specifications, the *ASCQM* standard contains Detection Patterns that are not applicable to all programming languages. When a pattern is not applicable, there are no Detection Pattern Occurrences to process.

7.15.3 Shared Pattern Considerations

Detection Patterns are occasionally shared between Weaknesses (e.g., among Weaknesses within the same parent-child groups of Weaknesses). Each unique Detection Pattern Occurrence would only be fixed only once. Therefore, each Detection Pattern Occurrence must be counted only once within the computation of Remediation Effort for a Quality Characteristic.

E.g. ASCQM Ban Use of Expired Pointer Detection Pattern supports Weaknesses CWE-825, CWE-119, CWE-416, and CWE-672, all of which are associated to Security quality characteristic. Therefore, when computing SREM, ASCQM Ban Use of Expired Pointer Detection Pattern Remediation Effort shall contribute only once.

7.15.4 Overlapping Pattern Considerations

Although designed to avoid functional overlap among Detection Patterns, there is at least one case where one Detection Pattern entirely overlaps another Detection Pattern: ASCQM Initialize Variables entirely overlaps ASCQM Initialize Variables before Use. That is, all Occurrences of ASCQM Initialize Variables before Use are also Occurrences of ASCQM Initialize Variables (the two Detection Patterns were defined to discriminate between two Weaknesses, CWE-456 Missing Initialization of a Variable and CWE-457 Use of Uninitialized Variable). Therefore, when the Remediation Effort of both Detection Patterns (ASCQM Initialize Variables and ASCQM Initialize Variables before Use) should be summed together, only the Remediation Effort of both ASCQM Initialize Variables is taken into account.

7.15.5 Measures' Specifications

MREM is an **smm:CollectiveMeasure that shall** sum the pattern-level Remediation Effort measure values from Sub-clause 7.13 (note that the **smm:MeasureRelationship** elements towards Detection Pattern level measures are not shown here)

<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.cat.280.category_remediation_effort" name="Weakness Category Maintainability Remediation Effort" unit="effort(minutes)" scope="toRevisionMeasurementScope" trait="RemediationEffortEstimating" category="id.cat.280" shortDescription="Effort to remove all occurrences of Maintainability weakness category (measured as the sum of remediation efforts of all contributing detection patterns)" accumulator="sum" baseMeasureTo="id.cat.280.category_remediation_effort_to_id.sfgd.102.pattern_remediation_effort id.cat.280.category_remediation_effort_to_id.sfgd.103.pattern_remediation_effort id.cat.280.category_remediation_effort_to_id.sfgd.104.pattern_remediation_effort id.cat.280.category_remediation_effort_to_id.sfgd.112.pattern_remediation_effort id.cat.280.category_remediation_effort_to_id.sfgd.113.pattern_remediation_effort id.cat.280.category_remediation_effort_to_id.sfgd.116.pattern_remediation_effort id.cat.280.category_remediation_effort_to_id.sfgd.123.pattern_remediation_effort id.cat.280.category_remediation_effort_to_id.sfgd.124.pattern_remediation_effort

id.cat.280.category_remediation_effort_to_id.sfgd.125.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.127.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.128.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.129.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.130.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.131.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.132.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.135.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.136.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.137.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.140.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.153.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.155.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.156.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.159.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.232.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.318.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.319.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.332.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.336.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.341.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.57.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.59.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.60.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.72.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.84.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.87.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.88.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.90.pattern_remediation_effort
id.cat.280.category_remediation_effort_to_id.sfgd.98.pattern_remediation_effort"/>

RREM is an **smm:CollectiveMeasure that shall** sum the pattern-level Remediation Effort measure values from Sub-clause 7.13 (note that the **smm:MeasureRelationship** elements towards pattern level measures are not shown here)

<measureElement xmi:type="smm:CollectiveMeasure"
xmi:id="id.cat.278.category_remediation_effort" name="Weakness Category Reliability Remediation Effort" unit="effort(minutes)" scope="toRevisionMeasurementScope" trait="RemediationEffortEstimating" category="id.cat.278" shortDescription="Effort to remove all occurrences of Reliability weakness category (measured as the sum of remediation efforts of all contributing detection patterns)" accumulator="sum"
baseMeasureTo="id.cat.278.category_remediation_effort_to_id.sfgd.106.pattern_remediation_effort id.cat.278.category_remediation_effort_to_id.sfgd.107.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.109.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.110.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.114.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.12.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.120.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.121.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.122.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.123.pattern_remediation_effort

id.cat.278.category_remediation_effort_to_id.sfgd.125.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.126.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.127.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.128.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.133.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.134.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.136.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.138.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.140.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.141.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.148.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.149.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.15.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.154.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.157.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.189.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.19.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.190.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.232.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.238.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.24.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.25.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.26.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.260.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.261.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.27.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.29.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.290.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.297.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.30.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.301.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.305.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.312.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.313.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.317.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.320.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.321.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.326.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.327.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.328.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.329.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.333.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.334.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.335.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.337.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.338.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.339.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.340.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.341.pattern_remediation_effort

id.cat.278.category_remediation_effort_to_id.sfgd.344.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.38.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.41.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.44.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.45.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.57.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.59.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.60.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.61.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.69.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.78.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.79.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.81.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.82.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.83.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.85.pattern_remediation_effort
id.cat.278.category_remediation_effort_to_id.sfgd.92.pattern_remediation_effort"/>

SREM is an **smm:CollectiveMeasure that shall** sum the pattern-level Remediation Effort measure values from Sub-clause 7.13 (note that the **smm:MeasureRelationship** elements towards pattern level measures are not shown here)

<measureElement xmi:type="smm:CollectiveMeasure"
xmi:id="id.cat.277.category_remediation_effort" name="Weakness Category Security Remediation Effort" unit="effort(minutes)" scope="toRevisionMeasurementScope" trait="RemediationEffortEstimating" category="id.cat.277" shortDescription="Effort to remove all occurrences of Security weakness category (measured as the sum of remediation efforts of all contributing detection patterns)" accumulator="sum"
baseMeasureTo="id.cat.277.category_remediation_effort_to_id.sfgd.106.pattern_remediation_effort id.cat.277.category_remediation_effort_to_id.sfgd.107.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.119.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.121.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.122.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.123.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.125.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.126.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.127.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.128.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.133.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.138.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.140.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.141.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.143.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.144.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.145.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.146.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.147.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.15.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.150.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.151.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.152.pattern_remediation_effort

id.cat.277.category_remediation_effort_to_id.sfgd.157.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.189.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.19.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.190.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.232.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.24.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.25.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.252.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.26.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.260.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.261.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.27.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.29.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.30.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.301.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.305.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.312.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.313.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.317.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.318.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.319.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.320.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.322.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.323.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.326.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.328.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.329.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.330.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.331.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.332.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.333.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.334.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.335.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.337.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.338.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.339.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.340.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.341.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.343.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.344.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.44.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.57.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.59.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.60.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.69.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.72.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.78.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.79.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.82.pattern_remediation_effort

id.cat.277.category_remediation_effort_to_id.sfgd.83.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.85.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.92.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.93.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.94.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.95.pattern_remediation_effort
id.cat.277.category_remediation_effort_to_id.sfgd.96.pattern_remediation_effort"/>

PEREM is an **smm:CollectiveMeasure that shall** sum the pattern-level Remediation Effort measure values from Sub-clause 7.13 (note that the **smm:MeasureRelationship** elements towards pattern level measures are not shown here)

measureElement xmi:type="smm:CollectiveMeasure"
xmi:id="id.cat.279.category_remediation_effort" name="Weakness Category Performance Efficiency Remediation Effort" unit="effort(minutes)" scope="toRevisionMeasurementScope" trait="RemediationEffortEstimating" category="id.cat.279" shortDescription="Effort to remove all occurrences of Performance Efficiency weakness category (measured as the sum of remediation efforts of all contributing detection patterns)" accumulator="sum"
baseMeasureTo="id.cat.279.category_remediation_effort_to_id.sfgd.100.pattern_remediation_effort id.cat.279.category_remediation_effort_to_id.sfgd.101.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.105.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.106.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.108.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.111.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.117.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.118.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.125.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.127.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.128.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.138.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.139.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.140.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.157.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.260.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.335.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.344.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.69.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.78.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.91.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.97.pattern_remediation_effort
id.cat.279.category_remediation_effort_to_id.sfgd.99.pattern_remediation_effort"/>

7.16 Quantification of Remediation Effort at the Software Level (ATDM)

7.16.1 Calculating Software Remediation Effort

The Automated Technical Debt Measure (ATDM) shall be calculated by summing the Remediation Efforts of all Detection Patterns in the *ASCQM* standard. The pattern applicability considerations, shared pattern considerations, and overlapping pattern considerations (sub-clauses 7.15.2 to 7.15.4) that applied to calculating Remediation Effort at the Quality Characteristic level also apply to calculating Remediation Effort at the software level.

7.16.2 Measure Specifications

- *ATDM* is an **smm:CollectiveMeasure** that shall sum the pattern-level Remediation Effort measure values from Sub-clause 7.15 (note that the **smm:MeasureRelationship** elements towards pattern level measures are not shown here)

  - **<measureElement xmi:type="smm:CollectiveMeasure" xmi:id="id.atdm_remediation_effort" name="Automated Technical Debt Remediation Effort" unit="effort(minutes)" scope="toRevisionMeasurementScope" trait="RemediationEffortEstimating" shortDescription="Automated Technical Debt weakness category (measured as the sum of remediation efforts of all contributing detection patterns)" accumulator="sum" baseMeasureTo="id.atdm_remediation_effort_to_id.sfgd.100.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.101.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.102.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.103.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.104.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.105.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.106.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.107.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.108.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.109.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.110.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.111.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.112.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.113.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.114.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.116.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.117.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.118.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.119.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.12.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.120.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.121.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.122.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.123.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.124.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.125.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.126.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.127.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.128.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.129.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.130.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.131.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.132.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.133.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.134.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.135.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.136.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.137.pattern_remediation_effort id.atdm_remediation_effort_to_id.sfgd.138.pattern_remediation_effort**

id.atdm_remediation_effort_to_id.sfgd.139.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.140.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.141.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.143.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.144.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.145.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.146.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.147.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.148.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.149.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.15.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.150.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.151.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.152.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.153.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.154.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.155.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.156.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.157.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.159.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.189.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.19.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.190.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.232.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.238.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.24.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.25.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.252.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.26.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.260.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.261.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.27.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.29.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.290.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.297.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.30.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.301.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.305.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.312.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.313.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.317.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.318.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.319.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.320.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.321.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.322.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.323.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.326.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.327.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.328.pattern_remediation_effort

id.atdm_remediation_effort_to_id.sfgd.329.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.330.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.331.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.332.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.333.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.334.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.335.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.336.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.337.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.338.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.339.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.34.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.340.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.341.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.343.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.344.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.38.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.41.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.44.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.45.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.57.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.59.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.60.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.61.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.69.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.72.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.78.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.79.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.81.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.82.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.83.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.84.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.85.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.87.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.88.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.90.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.91.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.92.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.93.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.94.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.95.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.96.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.97.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.98.pattern_remediation_effort
id.atdm_remediation_effort_to_id.sfgd.99.pattern_remediation_effort"/>

7.17 ASCQM Unadjusted Remediation Effort Configuration

Table 5 lists the Unadjusted Remediation Effort values to be used with each ASCQM Weakness in Sub-clause 6.2 and its associated Detection Patterns.

Table 5. Unadjusted Remediation Effort for Each ISO 5055 Detection Pattern

Detection Pattern	Unadjusted Remediation Effort (minutes)
ASCQM Ban Allocation of Memory with Null Size	33
ASCQM Ban Assignment Operation Inside Logic Blocks	33
ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size	79
ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation	79
ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value	79
ASCQM Ban Circular Dependencies between Modules	100
ASCQM Ban Comma Operator from Delete Statement	79
ASCQM Ban Comparison Expression Outside Logic Blocks	33
ASCQM Ban Control Flow Transfer	100
ASCQM Ban Conversion References to Child Class	79
ASCQM Ban Creation of Lock On Inappropriate Object Type	140
ASCQM Ban Creation of Lock On Non-Final Object	140
ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data	140
ASCQM Ban Delete of VOID Pointer	140
ASCQM Ban Double Free On Pointers	57
ASCQM Ban Double Release of Resource	100
ASCQM Ban Empty Exception Block	57
ASCQM Ban Exception Definition without Ever Throwing It	100
ASCQM Ban Excessive Complexity of Data Resource Access	179
ASCQM Ban Excessive Number of Children	140
ASCQM Ban Excessive Number of Concrete Implementations to Inherit From	100
ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code	140
ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure	179
ASCQM Ban Excessive Number of Index on Columns of Large Tables	140
ASCQM Ban Excessive Number of Inheritance Levels	179
ASCQM Ban Excessive Size of Index on Columns of Large Tables	100
ASCQM Ban Expensive Operations in Loops	179
ASCQM Ban File Creation with Default Permissions	79
ASCQM Ban Free Operation on Pointer Received as Parameter	79
ASCQM Ban Hard-Coded Literals used to Connect to Resource	57
ASCQM Ban Hard-Coded Literals used to Initialize Variables	46
ASCQM Ban Incompatible Lock Acquisition Sequences	179
ASCQM Ban Incorrect Float Number Comparison	57
ASCQM Ban Incorrect Joint Comparison	57
ASCQM Ban Incorrect Numeric Conversion of Return Value	57
ASCQM Ban Incorrect Numeric Implicit Conversion	46
ASCQM Ban Incorrect Object Comparison	33

ASCQM Ban Incorrect String Comparison	33
ASCQM Ban Incorrect Synchronization Mechanisms	121
ASCQM Ban Incorrect Type Conversion	33
ASCQM Ban Incrementral Creation of Immutable Data	57
ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities	57
ASCQM Ban Logical Dead Code	100
ASCQM Ban Logical Operation with a Constant Operand	33
ASCQM Ban Loop Value Update within Incremental and Decremental Loop	57
ASCQM Ban Non-Final Static Data in Multi-Threaded Context	100
ASCQM Ban Non-Serializable Elements in Serializable Objects	79
ASCQM Ban Not Operator On Non-Boolean Operand Of Comparison Operation	57
ASCQM Ban Not Operator On Operand Of Bitwise Operation	57
ASCQM Ban Reading and Writing the Same Variable Used as Assignment Value	140
ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context	100
ASCQM Ban Return of Local Variable Address	140
ASCQM Ban Self Assignment	79
ASCQM Ban Self Destruction	79
ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock	100
ASCQM Ban Sleep Between Lock Acquisition and Release	140
ASCQM Ban Static Non-Final Data Element Outside Singleton	79
ASCQM Ban Storage of Local Variable Address in Global Variable	140
ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities	57
ASCQM Ban Switch in Switch Statement	179
ASCQM Ban Unintended Paths	100
ASCQM Ban Unmodified Loop Variable Within Loop	57
ASCQM Ban Unreferenced Dead Code	100
ASCQM Ban Usage of Data Elements from Other Classes	57
ASCQM Ban Use of Deprecated Libraries	79
ASCQM Ban Use of Expired Pointer	79
ASCQM Ban Use of Expired Resource	100
ASCQM Ban Use of Prohibited Low-Level Resource Management Functionality	140
ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues	179
ASCQM Ban Useless Handling of Exceptions	57
ASCQM Ban Variable Increment or Decrement Operation in Operations using the Same Variable	140
ASCQM Ban While TRUE Loop Without Path To Break	57
ASCQM Catch Exceptions	33
ASCQM Check and Handle ZERO Value before Use as Divisor	33
ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use	46
ASCQM Check Index of Array Access	46
ASCQM Check Input of Memory Allocation Primitives	79
ASCQM Check Input of Memory Manipulation Primitives	57

ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities	57
ASCQM Check NULL Pointer Value before Use	46
ASCQM Check Offset used in Pointer Arithmetic	100
ASCQM Check Return Value of Resource Operations Immediately	57
ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context	140
ASCQM Handle Return Value of Must Check Operations	57
ASCQM Handle Return Value of Resource Operations	57
ASCQM Implement Copy Constructor for Class With Pointer Resource	79
ASCQM Implement Correct Object Comparison Operations	79
ASCQM Implement Index Required by Query on Large Tables	140
ASCQM Implement Required Operations for Manual Resource Management	57
ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor	57
ASCQM Implement Virtual Destructor for Classes with Virtual Methods	57
ASCQM Implement Virtual Destructor for Parent Classes	46
ASCQM Initialize Pointers before Use	33
ASCQM Initialize Resource before Use	57
ASCQM Initialize Variables	33
ASCQM Initialize Variables before Use	33
ASCQM Limit Algorithmic Complexity via Cyclomatic Complexity Value	79
ASCQM Limit Algorithmic Complexity via Essential Complexity Value	179
ASCQM Limit Algorithmic Complexity via Module Design Complexity Value	179
ASCQM Limit Number of Aggregated Non-Primitive Data Types	100
ASCQM Limit Number of Data Access	100
ASCQM Limit Number of Outward Calls	140
ASCQM Limit Number of Parameters	100
ASCQM Limit Size of Operations Code	179
ASCQM Limit Volume of Commented-Out Code	57
ASCQM Limit Volume of Similar Code	79
ASCQM Log Caught Security Exceptions	79
ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls	46
ASCQM NULL Terminate Output Of String Manipulation Primitives	33
ASCQM Release File Resource after Use in Class	79
ASCQM Release File Resource after Use in Operation	79
ASCQM Release in Destructor Memory Allocated in Constructor	79
ASCQM Release Lock After Use	140
ASCQM Release Memory After Use	79
ASCQM Release Memory after Use with Correct Operation	79
ASCQM Release Platform Resource after Use	79
ASCQM Sanitize Stored Input used in User Output	140
ASCQM Sanitize User Input used as Array Index	33
ASCQM Sanitize User Input used as Pointer	100

ASCQM Sanitize User Input used as Serialized Object	79
ASCQM Sanitize User Input used as String Format	79
ASCQM Sanitize User Input used in Document Manipulation Expression	79
ASCQM Sanitize User Input used in Document Navigation Expression	57
ASCQM Sanitize User Input used in Expression Language Statement	79
ASCQM Sanitize User Input used in Loop Condition	57
ASCQM Sanitize User Input used in Path Manipulation	46
ASCQM Sanitize User Input used in SQL Access	79
ASCQM Sanitize User Input used in System Command	79
ASCQM Sanitize User Input used in User Output	140
ASCQM Sanitize User Input used to access Directory Resources	100
ASCQM Secure Use of Unsafe XML Processing with Secure Parser	57
ASCQM Secure XML Parsing with Secure Options	57
ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context	79
ASCQM Use Break in Switch Statement	46
ASCQM Use Default Case in Switch Statement	33

Table 6 summarizes the Unadjusted Remediation Effort values for use with each ASCQM Weakness based on their associated Detection Patterns. For each ASCQM Weakness, it also lists the Quality Characteristics to which the Weakness is a member. When the Effort is listed as a range, the choice of Unadjusted Remediation Effort depends on which Detection Pattern must be remediated to correct the Weakness.

Table 6. Unadjusted Remediation Effort and Quality Characteristic Membership for Each ISO 5055 Weakness

CWE #	Weakness title	Quality Characteristic Membership	Unadjusted Remediation Effort (hours)
22	**Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')**	Maintainability	**0.8**
23	**Relative Path Traversal**	Security	**0.8**
36	**Absolute Path Traversal**	Security	**0.8**
77	**Improper Neutralization of Special Elements used in a Command ('Command Injection')**	Security	**1.3**
78	**Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')**	Security	**1.3**
79	**Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')**	Security	**2.3**
88	**Argument Injection or Modification**	Security	**1.3**
89	**Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')**	Security	**1.3**
90	**Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')**	Security	**1.7**

91	**XML Injection (aka Blind XPath Injection)**	Security	**0.9 – 1.3**
99	**Improper Control of Resource Identifiers ('Resource injection')**	Security	**0.8**
119	**Improper Restriction of Operations within the Bounds of a Memory Buffer**	Reliability Security	**0.6 – 1.7**
120	**Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')**	Reliability Security	**0.9**
123	**Write-what-where condition**	Reliability Security	**0.9**
125	**Out-of-bounds Read**	Reliability Security	**0.8**
129	**Improper Validation of Array Index**	Security	**0.6**
130	**Improper Handling of Length Parameter Inconsistency**	Reliability Security	**0.8**
131	**Incorrect Calculation of Buffer Size**	Reliability Security	**1.3**
134	**Use of Externally Controlled Format String**	Security	**1.3**
170	**Improper Null Termination**	Security	**0.6**
194	**Unexpected Sign Extension**	Reliability Security	**0.8**
195	**Signed to Unsigned Conversion Error**	Reliability Security	**0.8**
196	**Unsigned to Signed Conversion Error**	Reliability Security	**0.8**
197	**Numeric Truncation Error**	Reliability	**0.8**
248	**Uncaught Exception**	Reliability	**0.6**
252	**Unchecked Return Value**	Reliability Security	**0.9**
259	**Use of Hard-coded Password**	Security	**0.9**
321	**Use of Hard-coded Cryptographic Key**	Security	**0.9**
366	**Race Condition within a Thread**	Reliability Security	**2.3**
369	**Divide By Zero**	Reliability Security	**0.6**
390	**Detection of Error Condition Without Action**	Reliability	**0.9**
391	**Unchecked Error Condition**	Reliability	**0.9**
392	**Missing Report of Error Condition**	Reliability	**0.9**

394	**Unexpected Status Code or Return Value**	Reliability	**0.9**
401	**Improper Release of Memory Before Removing Last Reference ('Memory Leak')**	Reliability Security Performance	**0.9 – 1.3**
404	**Improper Resource Shutdown or Release**	Reliability Security Performance	**0.8 – 1.3**
407	**Algorithmic Complexity**	Maintainability	**1.3 – 3.0**
415	**Double Free**	Reliability Security	**0.9**
416	**Use After Free**	Reliability Security	**1.3**
424	**Improper Protection of Alternate Path**	Reliability Security Performance	**1.7**
434	**Unrestricted Upload of File with Dangerous Type**	Security	**0.8**
456	**Missing Initialization of a Variable**	Reliability Security	**0.6**
457	**Use of uninitialized variable**	Reliability Security	**0.6**
459	**Incomplete Cleanup**	Reliability	**1.3**
476	**NULL Pointer Dereference**	Reliability	**0.8**
477	**Use of Obsolete Function**	Security	**1.3**
478	**Missing Default Case in Switch Statement**	Maintainability	**0.6**
480	**Use of Incorrect Operator**	Reliability Security Maintainability	**0.6**
484	**Omitted Break Statement in Switch**	Reliability Maintainability	**0.8**
502	**Deserialization of Untrusted Data**	Security	**1.3**
543	**Use of Singleton Pattern Without Synchronization in a Multithreaded Context**	Reliability Security	**1.3 – 1.7**
561	**Dead code**	Maintainability	**1.7**
562	**Return of Stack Variable Address**	Reliability	**2.3**
564	**SQL Injection: Hibernate**	Security	**1.3**
567	**Unsynchronized Access to Shared Data in a Multithreaded Context**	Reliability Security	**1.7 – 2.3**

570	**Expression is Always False**	Security Maintainability	**0.8**
571	**Expression Is Always True**	Security Maintainability	**0.8**
595	**Comparison of Object References Instead of Object Contents**	Reliability	**0.6 – 1.3**
597	**Use of Wrong Operator in String Comparison**	Reliability	**0.6**
606	**Unchecked Input for Loop Condition**	Security	**0.9**
611	**Improper Restriction of XML External Entity Reference ('XXE')**	Security	**0.9**
643	**Improper Neutralization of Data within XPath Expressions ('XPath Injection')**	Security	**0.9**
652	**Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')**	Security	**1.3**
662	**Improper Synchronization**	Reliability Security	**1.3 – 3.0**
665	**Improper Initialization**	Reliability Security	**0.6 – 1.3**
667	**Improper Locking**	Reliability Security	**1.7 – 2.3**
672	**Operation on a Resource after Expiration or Release**	Reliability Security	**0.9 – 1.7**
681	**Incorrect Conversion between Numeric Types**	Reliability Security	**0.8**
682	**Incorrect Calculation**	Reliability Security	**0.6 – 1.3**
703	**Improper Check or Handling of Exceptional Conditions**	Reliability	**0.6 – 0.9**
704	**Incorrect Type Conversion or Cast**	Reliability	**0.6**
732	**Incorrect Permission Assignment for Critical Resource**	Security	**1.3**
758	**Reliance on Undefined, Unspecified, or Implementation-Defined Behavior**	Reliability	**2.3**
764	**Multiple Locks of a Critical Resource**	Reliability	**1.7**
772	**Missing Release of Resource after Effective Lifetime**	Reliability Security Performance	**1.3**
775	**Missing Release of File Descriptor or Handle after Effective Lifetime**	Reliability Security Performance	**1.3**

778	**Insufficient Logging**	Security	**1.3**
783	**Operator Precedence Logic Error**	Security Maintainability	**0.9**
786	**Access of Memory Location Before Start of Buffer**	Reliability Security	**0.8 – 0.9**
787	**Out-of-bounds Write**	Reliability Security	**0.8 – 0.9**
788	**Access of Memory Location After End of Buffer**	Reliability Security	**0.8 – 0.9**
789	**Uncontrolled Memory Allocation**	Security	**0.6 – 1.3**
798	**Use of Hard-coded Credentials**	Security	**0.9**
805	**Buffer Access with Incorrect Length Value**	Reliability Security	**0.9**
820	**Missing Synchronization**	Reliability Security	**1.7**
821	**Incorrect Synchronization**	Reliability Security	**2.0**
822	**Untrusted Pointer Dereference**	Reliability Security	**1.7**
823	**Use of Out-of-range Pointer Offset**	Reliability Security	**1.7**
824	**Access of Uninitialized Pointer**	Reliability Security	**0.6**
825	**Expired Pointer Dereference**	Reliability Security	**1.3**
833	**Deadlock**	Reliability	**3.0**
835	**Loop with Unreachable Exit Condition ('Infinite Loop')**	Reliability Security	**0.9**
908	**Use of Uninitialized Resource**	Reliability	**0.9**
917	**Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')**	Security	**1.3**
1041	**Use of Redundant Code (Copy-Paste)**	Maintainability	**1.3**
1042	**Static Member Data Element outside of a Singleton Class Element**	Performance	**1.3**
1043	**Data Element Aggregating an Excessively Large Number of Non-Primitive Elements**	Performance	**1.7**
1045	**Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor**	Reliability Maintainability	**0.9**

1046	**Creation of Immutable Text Using String Concatenation**	Performance	**0.9**
1047	**Modules with Circular Dependencies**	Maintainability	**1.7**
1048	**Invokable Control Element with Large Number of Outward Calls (Excessive Coupling or Fan-out)**	Maintainability	**2.3**
1049	**Excessive Data Query Operations in a Large Data Table**	Performance	**3.0**
1050	**Excessive Platform Resource Consumption within a Loop**	Performance	**3.0**
1051	**Initialization with Hard-Coded Network Resource Configuration Data**	Reliability Maintainability	**0.9**
1052	**Excessive Use of Hard-Coded Literals in Initialization**	Maintainability	**0.8**
1054	**Invocation of a Control Element at an Unnecessarily Deep Horizontal Layer (Layer-skipping Call)**	Maintainability	**1.7**
1055	**Multiple Inheritance from Concrete Classes**	Maintainability	**1.7**
1057	**Data Access Operations Outside of Expected Data Manager Component**	Security Performance	**1.7**
1058	**Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element**	Reliability	**1.7**
1060	**Excessive Number of Inefficient Server-Side Data Accesses**	Performance	**3.0**
1062	**Parent Class Element with References to Child Class**	Maintainability	**1.3**
1064	**Invokable Control Element with Signature Containing an Excessive Number of Parameters**	Maintainability	**1.7**
1066	**Missing Serialization Control Element**	Reliability	**1.3**
1067	**Excessive Execution of Sequential Searches of Data Resource**	Performance	**2.3**
1070	**Serializable Data Element Containing non-Serializable Item Elements**	Reliability	**1.3**
1072	**Non-SQL Invokable Control Element with Excessive Number of Data Resource Accesses**	Performance	**2.3**
1073	**Large Data Table with Excessive Number of Indices**	Performance	**2.3**
1074	**Class with Excessively Deep Inheritance**	Maintainability	**3.0**

1075	**Unconditional Control Flow Transfer outside of Switch Block**	Maintainability	**1.7**
1077	**Floating Point Comparison with Incorrect Operator**	Reliability	**0.9**
1079	**Parent Class without Virtual Destructor Method**	Reliability Maintenance	**0.8**
1080	**Source Code File with Excessive Number of Lines of Code**	Maintainability	**3.0**
1082	**Class Instance Self Destruction Control Element**	Reliability	**1.3**
1083	**Data Access from Outside Designated Data Manager Component**	Reliability	**1.7**
1084	**Invokable Control Element with Excessive File or Data Access Operations**	Maintainability	**1.7**
1085	**Invokable Control Element with Excessive Volume of Commented-out Code**	Maintainability	**0.9**
1086	**Class with Excessive Number of Child Classes**	Maintainability	**2.3**
1087	**Class with Virtual Method without a Virtual Destructor**	Reliability Maintainability	**0.9**
1088	**Synchronous Access of Remote Resource without Timeout**	Reliability	**0.8**
1089	**Large Data Table with Excessive Number of Indices**	Performance	**2.3**
1090	**Method Containing Access of a Member Element from Another Class**	Maintainability	**0.9**
1091	**Use of Object without Invoking Destructor Method**	Performance	**0.9**
1094	**Excessive Index Range Scan for a Data Resource**	Performance	**1.7**
1095	**Loop Condition Value Update within the Loop**	Maintainability	**0.9**
1096	**Singleton Class Instance Creation without Proper Locking or Synchronization**	Reliability	**1.3**
1097	**Persistent Storable Data Element without Associated Comparison Control Element**	Reliability	**1.3**
1098	**Data Element containing Pointer Item without Proper Copy Control Element**	Reliability	**1.3**
1121	**Excessive McCabe Cyclomatic Complexity**	Maintainability	**1.3**

7.18 Output Generation

The last step of the automated process shall generate the output. The output shall be a human readable report that contains sufficient detail to answer the following questions:

- What is the amount of Automated Technical Debt (ATDM)?
- What is the amount of Remediation Effort required for each of the Quality Characteristic measures (Automated Maintainability/Reliability/Performance Efficiency/Security)?
- What is the amount of ATDM added or removed between two revisions?
- What is the amount of ATDM concentrated in any set of code elements?
- What are the exposure levels of individual occurrences in the ATDM?
- What are the assumptions used in calculating ATDM?

The generated output file format shall be a common text file format (e.g., .txt or .csv) to allow for importing to other tools such as Excel or a commercial software estimating package. The output shall include the following artifacts:

- At the measurement level
  - *ASCSM*, *ASCRM*, *ASCPEM*, and *ASCMM* measurement input
  - Remediation Effort configuration input (if not the default values)
  - *AEP Effort Complexity* measurement input (if not the default values)
- At the software revision level
  - *ATDM* value
  - *MREM*, *RREM*, *PEREM*, and *SREM* values
- At the weakness level, for all weaknesses
  - Weakness Remediation Effort values
- At the pattern level, for all patterns
  - Pattern Remediation Effort values
- At the Occurrence level, for all Occurrences of all patterns
  - Occurrence Remediation Effort values
  - Occurrence Adjustment Factor values
  - Occurrence Complexity and Exposure overhead average values
  - Occurrence Sharing Opportunity average values
  - Occurrence Technological Diversity values
  - Occurrence Evolution
- At the role level, for all Occurrences of all patterns
  - List of code elements implementing a role
  - Complexity of role implementation code elements
  - Concentration of role implementation code elements
  - Evolution of role implementation code elements
  - Direct and indirect Exposure of role implementation code elements

8 Automated Technical Debt Measure (ATDM) Usage Scenarios (Informative)

8.1 Risk Mitigation

The following scenarios illustrate ways in which the Automated Technical Debt Measure (ATDM) and Contextual measures can be used to help mitigate the risk of the Technical Debt associated with IT applications.

8.1.1 ATDM and Its Component Effort Values for MREM, RREM, PEREM, SREM

Action—Compare the ATDM value and individual ASCQM Quality Characteristic Remediation values (MREM, RREM, PEREM, SREM).

Interpretation—This comparison helps determine when the total Technical Debt measured in the ATDM value (normalized by size, if needed) is unequally distributed between Technical Debt Items associated with Security, Performance Efficiency, Maintainability, or Reliability.

8.1.2 Exposure

Action—Chart the Occurrences of Technical Debt Items by Exposure values to evaluate the breadth of Risk Exposure.

Interpretation—This distribution helps identify which Technical Debt Items possess the greatest breadth of connections to other code elements in the software. These Technical Debt Items usually possess the greatest risk in effort and cost to correct because of potential side effects that must be evaluated. There is also greater possible destabilization resulting from undetected side effects of Remediation activities.

8.1.3 Evolution

Action—Chart the ATDM value by the Evolution Occurrences across releases.

Interpretation—This distribution helps identify trends in the management of Technical Debt. For instance, how much legacy Technical Debt exists in an application, and how much is being added or corrected in each subsequent release. Evolution can also be used in analyzing trends in the operational risks and cost of ownership associated with the Technical Debt as it is measured across releases.

8.2 Priority Setting

The following scenarios illustrate the ways measures defined in ATDM specifications can be used to help setting priorities for remediating Technical Debt Items.

8.2.1 ATDM and its component effort values for MREM, RREM, PEREM, SREM

Action—Evaluate the ASCQM Quality Characteristic Remediation values (MREM, RREM, PEREM, SREM).

Interpretation—The relative amounts of Technical Debt indicated in the Quality Characteristic Remediation values can help prioritize and allocate resources among the Quality Characteristics for remediating Technical Debt Items.

8.2.2 Technological Diversity

Action—Chart Occurrences of Technical Debt Items by their Technological Diversity.

Interpretation—This distribution identifies Technical Debt Items:

- that may require synchronization between multiple teams involved in a Remediation because different sets of computational objects involved in the Occurrence are written in different languages and may be located in different code elements.
- that probably can be handled by a single team because only one language is involved.

8.2.3 Exposure

Action—Chart Occurrences of Technical Debt Items by the range of Exposure values.

Interpretation—This distribution helps identify Technical Debt Items with:

- the highest Risk Exposure and Fix Destabilization Exposure so they can be corrected first during the release development cycle to remove the most impacting issues with enough time before the release to handle potential side-effects of the fix.
- the highest Fix Destabilization Exposure but lower Risk Exposure that so they can be corrected next during the release development cycle to remove issues while there is enough time to handle potential side-effects of the fix.
- the lowest Fix Destabilization Exposure that are to be removed near the end of the release development cycle to remove issues without jeopardizing the stability of the release.

8.2.4 Evolution

Acton—Chart Occurrences of Technical Debt Items by the Evolution of each Occurrence.

Interpretation—This distribution helps identify added Technical Debt Items that should be removed first to avoid letting future enhancements build on top of them, making them more difficult to remove in the future and increasing their potential negative impacts.

8.3 Productivity Measurement

The following scenario illustrates the way ATDM measures can be used in productivity analysis.

8.3.1 Evolution

Action—Filter the Occurrences of Technical Debt Items that were “added” in their Evolution .

Interpretation—Adjust productivity figures for the current release by including the Remediation Effort of Detection Patterns implemented in the current release but not corrected until a future release. Remediation Effort passed to future revisions is often counted as new work rather than rework, thus inflating productivity numbers.

9 Contextual Technical Debt Measure (CTDM) Usage Scenarios (Informative)

The Contextual Technical Debt Measure (CTDM) is an alternative to the Automated Technical Debt Measure because it is adapted to the context of a specific organization or application. The adaptation process is multifaceted and concerns one or more of the following non-mutually exclusive aspects:

- the list of patterns to consider: a subset of the patterns from the *ASCQM* standard; or a set including Detection Patterns not included in the *ASCQM* standard.
- different values for Remediation Effort: different Unadjusted Remediation Effort values,
- the use of different formulas for Adjustment Factors, or their deactivation
- the use of additional Adjustment Factors.

However, these adjustments are incorporated at the expense of benchmarking, which cannot be accomplished with CTDM except among applications where the CTDM adjustments are identical.

The following sub-clauses illustrates sample variations regarding Adjustment Factors.

9.1 Technological Diversity

Action—Adjust the Technological Diversity Adjustment Factor to better reflect the organization's ability to deal with Occurrences involving multiple technologies.

Illustrations

1. Turn off (that is, ignore from computation) the Technological Diversity Adjustment Factor if the organization is organized around cross-technology teams.
2. Compute an alternative Technological Diversity penalty factor equal to the power of the number of distinct technologies, with a power value smaller than 1, to model a smooth coordination of different teams, and greater than 1, to model the infrequent involvement of different teams.

9.2 Exposure

Action—Adjust the Exposure Adjustment Factor to better reflect the organization's ability to avoid destabilization of the software via automated testing.

Illustrations

1. Turn off (that is, ignore from computation) the Exposure Adjustment Factor if the organization is so mature regarding automated non-regression testing that teams can update the code without fear of side effects.
2. Compute an alternative Exposure Adjustment Factor using one of the following formulas:
   - with an asymptote: max-1/(range number+1)^{power}
   - without an asymptote: (range number)^{power}
   - where range number is a logarithmic transformation of the Exposure values, to account for combinatorial nature of the Exposure and make them human-friendly: |log (Exposure + 1)|

9.3 Sharing Opportunity

Action—Adjust the Sharing Opportunity Adjustment Factor to better reflect the organization's strategy regarding the removal of Technical Debt Occurrences.

Illustration—Turn off (that is, ignore from computation) the Sharing Opportunity Adjustment Factor if the organization is willing to remove Occurrences at different times, that is, without achieving an economy of effort when removing multiple Occurrences concurrently in the same code element.

9.4 Evolution

9.4.1 Occurrence

Action—Adjust the Remediation Effort for a Technical Debt Item with an evolution Contextual measure to factor in the opportunity to remove an Occurrence more easily when it was injected into the software during the current release cycle.

Illustration—Consider an Occurrence evolution reward factor of .50 for added Occurrences.

9.4.2 Code Elements

Action—Adjust the Remediation Effort for a Technical Debt Item with an evolution Contextual measure to factor in the opportunity to remove an Occurrence more easily when the code elements involved were recently updated.

Illustration—Consider a code element evolution reward factor of .75 for updated code elements.

9.5 Limitation

The use of Adjustment Factors makes the measures evolve over time. As Occurrences age and code elements are modified, remediating them becomes more difficult. This difficulty is compounded if there is growth in the number of Technical Debt items over cycles and releases.

10 Technical Debt Value Communication (Informative)

The following scenarios illustrate ways in which the Automated Technical Debt Measure (ATDM) and the Contextual Technical Debt Measure (CTDM) can be used to help communicate about Technical Debt with non-technical audiences, facilitate acceptance, and reap the benefits of the Technical Debt metaphor.

10.1 Problem statement

ATDM and CTDM are estimating the effort to remove all Occurrences of the selected Detection Patterns (from *ASCQM* standard, or from a user-defined list).

First, this is equivalent to a strategy of zero tolerance to defects which may be too stringent (and very likely unnecessary) to implement in all applications, as well as too expensive due to the sheer number of Occurrences to remediate. This leads to Remediation Effort values so large they are difficult to accept (even if justifiable), ultimately creating a push back against the measurement program.

Second, there is conceptual debate about the content of Technical Debt. Some say Technical Debt should only account for items that organizations intend to remove. In other words, if organizations do not plan to completely remove all Occurrences of each pattern, they are not to be considered in the Technical Debt measurement. A more academic approach holds that Technical Debt is not about Weaknesses, but rather the cost to remediate flaws resulting from conscious sub-optimal design decisions made in a rush to get software delivered on schedule.

Third, some organizations manage quality objectives, such as internal or external Service Level Agreements. That is, they may define the number of issues that are considered acceptable. When quality objectives are set with a certain tolerance value, only the Occurrences whose removal is needed to reach the target level of tolerance will be effectively removed; the remaining Occurrences will remain for lack of incentive to do so. In these situations, the Technical Debt values that are meaningful for the management are the estimations of the effort and cost to reach target values as re[resented in CDTM (as opposed to the estimation of the effort and cost to get the total absence of Occurrences).

10.2 Recommended Approach

10.2.1 When Quality Objectives Are Set

CISQ recommends the computation of the amount of Automated Technical Debt Measure that is required to reach quality objectives that are set for each application.

As the scope of the ATDM measure is adjusted with contextual information, this computation should be exposed labeled a Contextual Technical Debt Measure to avoid confusion.

The immediate benefits of this approach are:

1. a more relevant value, because it would be aligned with organization's existing management practices, as opposed to a value relative to a hypothetical "zero tolerance" situation,
2. a more acceptable value, because it would be smaller, having filtered out effort and cost amounts that are not ultimately applicable.

10.2.2 When Quality Objectives Are Not Set

In case no quality objectives are set, CISQ recommends the computation of the amount of Automated Technical Debt Measure required to reach arbitrary yet meaningful quality levels, such as sigma levels (e.g., Occurrences per million lines of code).

The immediate benefits are:

1. a perspective on quality levels, especially as there are no objectives set, to educate and help justify quality improvement initiatives (e.g., showing a plan to reach a sigma level 3 can resonate with non-technical management audience familiar with 6σ concepts)
2. a more acceptable value, because it would be smaller, based on remediating a smaller set of Occurrences. Removing all Occurrences would be unrealistic when dealing with an application for which there are no quality objectives.

10.3 Limitations

10.3.1 Benchmarking

Adjustments regarding tolerances are incorporated at the expense of benchmarking, which cannot be accomplished with CTDM except among applications where the CTDM adjustments are identical or acceptably different.

"Acceptably different" means there are differences in the adjustment criteria but that the organization is accepting and adhering to these differences and their impact on the way to interpret the results.

As an example, if two applications are assigned different tolerance levels, the organization must use the CTDM measures carefully. The measured values shall not be used to compare the Technical Debt for these two applications, but they shall be used to compare the distance to their respective quality objectives, using the Technical Debt metaphor.

10.3.2 Value Range

As soon as a tolerance level is not zero, some Occurrences will be allowed to remain in the software.

Each Detection Pattern Occurrence is assigned the same Unadjusted Remediation Effort. However, when the Adjustment Factors are applied, the Adjusted Remediation Effort will likely differ.

Therefore, the effort required to remove enough Occurrences to reach the quality objective for this Detection Pattern becomes a value range, with a minimum value obtained by targeting the Occurrences with the smaller adjusted Remediation Effort values, and with a maximum value obtained by targeting the Occurrences with the largest adjusted Remediation Effort values.

Annex A: Consortium for IT Software Quality (CISQ)

The purpose of the Consortium for IT Software Quality (CISQ) is to develop specifications for automated measures of software quality characteristics taken on source code. These measures were designed to provide international standards for measuring software structural quality that can be used by IT organizations, IT service providers, and software vendors in contracting, developing, testing, accepting, and deploying IT software applications. Executives from the member companies that joined CISQ prioritized the quality characteristics of Reliability, Security, Performance Efficiency, and Maintainability to be developed as measurement specifications.

CISQ strives to maintain consistency with ISO/IEC standards to the extent possible, and in particular with the ISO/IEC 25000 series that replaces ISO/IEC 9126 and defines quality measures for software systems. In order to maintain consistency with the quality model presented in ISO/IEC 25010, software quality characteristics are defined for the purpose of this specification as attributes that can be measured from the static properties of software and can be related to the dynamic properties of a computer system as affected by its software. However, the 25000 series, and in particular ISO/IEC 25023 which elaborates quality characteristic measures, does not define these measures at the source code level. Thus, this and other CISQ quality characteristic specifications supplement ISO/IEC 25023 by providing a deeper level of software measurement, one that is rooted in measuring software attributes in the source code.

Companies interested in joining CISQ held executive forums in Frankfurt, Germany; Arlington, VA; and Bangalore, India to set strategy and direction for the consortium. In these forums four quality characteristics were selected as the most important targets for automation—reliability, security, performance efficiency, and maintainability. These attributes cover four of the eight quality characteristics described in ISO/IEC 25010.

The Consortium for IT Software Quality (CISQ), a consortium managed by OMG, was formed in 2010 to create international standards for automating measures of size and structural quality characteristics from source code. These measures are intended for use by IT organizations, IT service providers, and software vendors in contracting, developing, testing, accepting, and deploying software systems. Executives from the member companies that joined CISQ prioritized Reliability, Security, Performance Efficiency, and Maintainability as the initial structural quality measures to be specified.

An international team of experts drawn from CISQ's 24 original companies formed into working groups to define CISQ measures. Weaknesses that had a high probability of causing reliability, security, performance efficiency, or maintainability problems were selected for inclusion in the four measures. The original CISQ members included IT departments in Fortune 200 companies, system integrators/ outsourcers, and vendors that provide quality-related products and services to the IT market. The experts met several times per year for two years in the US, France, and India to develop a broad list of candidate weaknesses. This list was pared down to a set of weaknesses they believed had to be remediated to avoid serious operational or cost problems. These 86 weaknesses became the foundation of the original specifications of the automated source code measures for Reliability, Security, Performance Efficiency, and Maintainability.