Date: June 11, 2015

Automated Source Code Performance Efficiency Measure

Beta 2

OMG Document Number: **ptc/2015-06-11**

Standard document URL:
http://www.omg.org/spec/ASCPEM/20150611/AutomatedSourceCodePerformanceEfficiencyMeasure

Associated files: Normative:

http://www.omg.org/spec/ASCPEM/20141110/AutomatedSourceCodeCISQTop15PerformanceEfficiencyMeasureSPMS.xmi

http://www.omg.org/spec/ASCPEM/20141110/AutomatedSourceCodeCISQTop15PerformanceEfficiencyMeasureSMM.smm

---

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group (OMG) 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 Consortium for IT Software Quality and it parent, 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

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. CISQ AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL CISDQ, THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE,

INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.287-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.287-19 or as specified in 48 C.F.R. 287-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 250 First Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

IMM®, MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

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 *http:// www.omg.org*, under Documents, Report a Bug/Issue (*http://www.omg.org/report_issue.htm*).

Table of Contents

Preface

About the Object Management Group

OMG

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

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

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

OMG Specifications

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

Business Modeling Specifications Middleware Specifications

- CORBA/IIOP
- Data Distribution Services
- Specialized CORBA

IDL/Language Mapping Specifications

Modeling and Metadata Specifications

- UML, MOF, CWM, XMI
- UML Profile

Modernization Specifications

Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications

- CORBAServices
- CORBAFacilities

CORBA Embedded Intelligence Specifications

CORBA Security Specifications

OMG Domain Specifications

Signal and Image Processing Specifications

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

OMG Headquarters
109 Highland Avenue
Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

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

Issues
The reader is encouraged to report and technical or editing issues/problems with this specification to **http://www.omg.org**

1. Scope

1.1 Purpose

The purpose of this specification is to establish a standard measure of Performance Efficiency based on detecting violations of good architectural and coding practices that could result in inefficient operation such as performance degradation or excessive use of processor resources. Establishing a standard for this measure is important because such measures are being used in outsourcing and system development contracts without having an approved international standard to reference. They are also critical to other software-intensive OMG initiatives such as The Internet of Things. The Consortium for IT Software Quality (CISQ) was formed as a special interest group of OMG to create specifications for automating standard measures of software quality attributes and submit them to OMG for approval.

1.2 CISQ Background

This specification defines a method for automating the measurement of Performance Efficiency from violations of architectural and coding practice that affect an application's performance and resource usage. The violations included in the CISQ measure were selected from a large set of candidate violations related to Performance Efficiency issues. The final set of violations were chosen through a voting process among CISQ member organizations that resulted in a limited set of violations that member organizations believed were sufficiently severe that they had to be remediated. This process will be described more fully in a subsequent sub-clause.

1.2 Overview of Software Quality Characteristic Measurement

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

Violations of Good Architectural and Coding Practice—The most recent advance in measuring the structural quality of software is based on the analysis and measurement of violations of good architectural and coding practice that can be detected by statically analyzing the source code. The CWE/SANS 25 and OWASP Top Ten lists of security weaknesses are examples of this approach. These lists are drawn from the Common Weakness Enumeration (CWE) repository maintained by MITRE Corporation. CWE contains descriptions of over 800 weaknesses that represent violations of good architectural and coding practice in software that can be exploited to gain unauthorized entry into a system. The Software Assurance

community has been a leader in this area of measurement by championing the detection of code weaknesses as a way of improving one aspect of software quality—software security.

Unfortunately there are no equivalent repositories of weaknesses for Reliability, Performance Efficiency, or Maintainability. Knowledge of these weaknesses is spread across software engineering textbooks, expert blogs, and information sharing sites such as github. The CISQ measure for Performance Efficiency can fill the void for a consensus body of knowledge about the most egregious Performance Efficiency problems that should be detected and remediated in source code. Currently, no standards or guidelines have been developed for calculating component or application-level Performance Efficiency measures that aggregate weaknesses detected through static code analysis into application-level Performance Efficiency measures. CISQ will be providing recommendations for these aggregation and scaling techniques. However, these techniques are not part of this standard since different measurement objectives are best served by different scoring techniques.

Using violations of good architectural and coding practices in software quality metrics presents several challenges for establishing baselines. Growth in the number of unique violations to be detected could continually raise the bar for measuring quality, reducing the validity of baseline comparisons. Further, different vendors will detect different sets of violations, making comparisons difficult across commercial software quality measurement offerings. One solution to this problem is to create a stable list of violations that are used for computing a baseline for each quality characteristic. The Automated Source Code Performance Efficiency Measure was developed by a team of industry experts to form the basis for a stable baseline measure.

1.3 Development of the Automated Source Code Performance Efficiency Measure

The original 24 CISQ member companies provided experts to working groups whose charter was to define CISQ measures. Violations of good architectural and coding practice that a high probability of causing Performance Efficiency problems were selected by an international team of experts drawn from the 24 organizations that joined CISQ in 2010. These organizations 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 Performance Efficiency weaknesses and then pare it down to a set they felt had to be remediated to avoid serious operational problems.

The work group began by defining Performance Efficiency issues, quality rules for avoiding these issues, and measures based on counting violations of these rules. They developed lists of issues and quality rules by drawing information from company defect logs, their career experience in different environments, and industry sources such as books and blogs. In order to reduce the work group's initial list to a critical set of Performance Efficiency violations, work group members individually evaluated the severity of each violation. High severity violations were judged to be those that must be fixed in a future release because of their operational risk or cost impact. The work group went through several rounds of eliminating lower severity violations and re-rating the severity of remaining violations until a final list was established as the quality measure elements to be incorporated into this specification.

1.4 Structure of the Automated Source Code Performance Efficiency Measure

ISO/IEC 25010 defines a quality characteristic as being composed from several quality sub-characteristics. This framework for software product quality is presented in Figure 1 for the eight quality characteristics presented in 25010. The quality characteristics and their sub-characteristics selected for source code measurement by CISQ are indicated in blue.

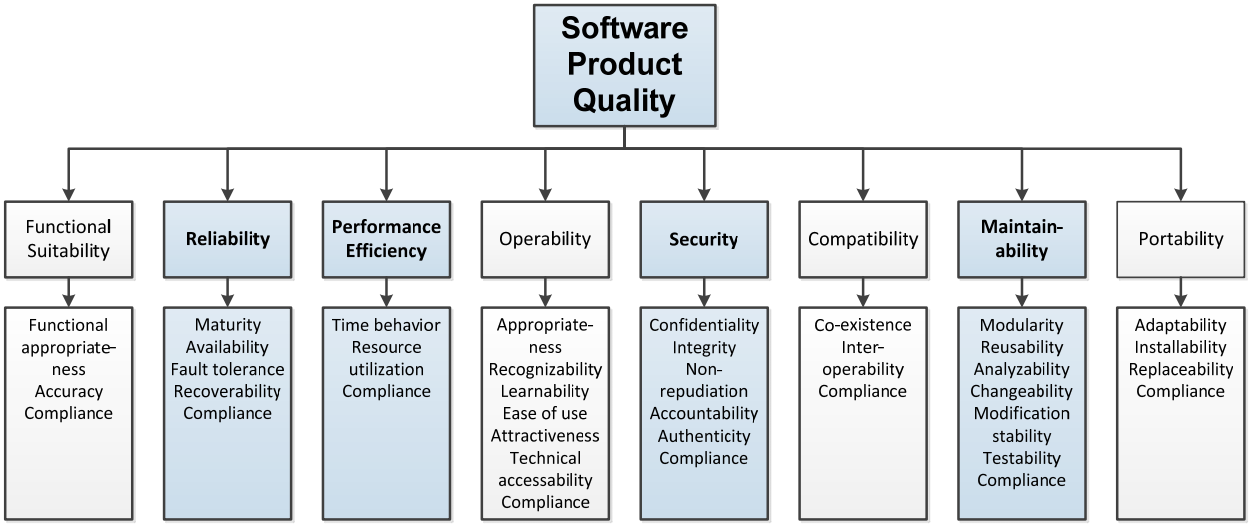

Figure 1. Software Quality Characteristics from ISO/IEC 25010 with CISQ focal areas highlighted.

ISO/IEC 25023 establishes a framework of software quality characteristic measures wherein each quality sub-characteristic consists of a collection of quality attributes that can be quantified as quality measure elements. A quality measure element quantifies a unitary measurable attribute of software, such as the violation of a quality rule. Figure 2 presents an example of the ISO/IEC 25023 quality measurement framework using a partial decomposition for the Automated Source Code Performance Efficiency Measure.

The non-normative portion of this specification begins by listing the Performance Efficiency issues that can plague software developed with poor architectural and coding practices. Quality rules written as architectural or coding practices are conventions that avoided the problem described in the Performance Efficiency issue. These quality rules were then transformed into software quality measure elements by counting violations of these architectural and coding practices and conventions.

The normative portion of this specification represents each quality measure element developed from a Performance Efficiency rule using the Structured Patterns Meta-model Standard (SPMS). The code-based elements in these patterns are represented in the Knowledge Discovery Meta-model (KDM). The calculation of the Automated Source Code Performance Efficiency Measure from its quality measure elements is then represented using the Structured Metrics Meta-model (SMM). This calculation is presented as the simple sum of quality measure elements without being adjusted by a weighting scheme.

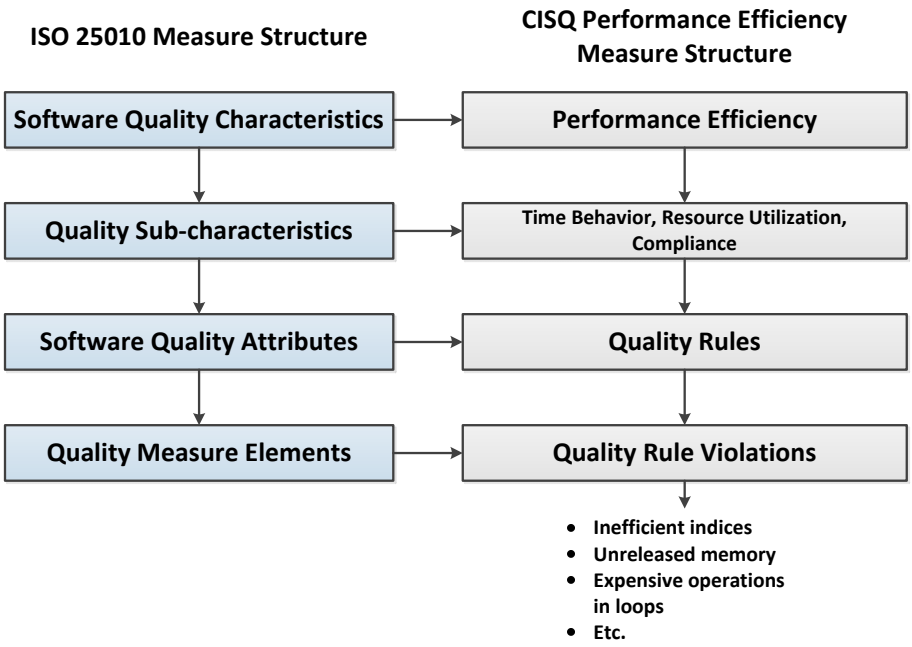

Figure 2. ISO/IEC 25010 Framework for Software Quality Characteristics Measurement

There are several weighting schemes that can be applied in aggregating violation counts into structural quality measures. The most effective weighting often depends on the measure's use such as assessing operational risk or estimating maintenance costs. The quality measure elements included in this specification were considered to be severe violations of secure architectural and coding practices that would need to be remediated. Therefore, weightings based on severity would add little useful information to the measure since the variance among weights would be small. In order to support benchmarking among applications, this specification includes a measure of the violation density. This measure is created by dividing the total number of violations detected by a count of Automated Function Points (Object Management Group, 2014).

1.5 Using and Improving This Measure

The Automated Source Code Performance Efficiency Measure is a correlated measure rather than an absolute measure. That is, since it does not measure all possible Performance Efficiency-related weaknesses it does not provide an absolute measure of Performance Efficiency. However, since it includes counts of what industry experts considered high severity Performance Efficiency weaknesses, it provides a strong indicator of Performance Efficiency that will be highly correlated with the absolute Performance Efficiency of a software system and with the probability that it can experience outages, data corruption, and related problems.

Since the impact and frequency of specific violations in the Automated Source Code Performance Efficiency Measure could change over time, this approach allows specific violations to be included, excluded, amplified, or diminished over time in order to support the most effective benchmarking, diagnostic, and predictive use. This specification will be adjusted through controlled OMG specification revision processes to reflect changes in Performance Efficiency engineering while retaining the ability to

compare baselines. Vendors of static analysis and measurement technology can compute this standard baseline measure, as well as their own extended measures that include other Performance Efficiency weaknesses not included as measure elements in this specification.

2. Conformance

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

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

3. Normative References

3.1 Normative

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.

- Structured Patterns Meta-model Standard, admtf/14-02-01
- Knowledge Discovery Meta-model, version 1.3 (KDM), formal/2011-08-04
- Structured Metrics Meta-model, version 1.0 (SMM), formal/2012-01-05
- MOF/XMI Mapping, version 2.4.1 (XMI), formal/2011-08-09
- Automated Function Points (AFP), formal/2014-01-03
- ISO/IEC 25010 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models

4. Terms and Definitions

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

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

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

Internal Software Quality—the degree to which a set of static attributes of a software product satisfy stated and implied needs for the software product to be used under specified conditions. This will be referred to as software structural quality, or simply structural quality in this specification. (ISO/IEC 25010)

Performance Efficiency—performance relative to the amount of resources or time used under stated conditions. (ISO/IEC 25010)

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

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

Software Product Quality Model—a model that categorizes product quality properties into eight characteristics (functional suitability, reliability, Performance Efficiency, usability, security, compatibility, maintainability and portability). Each characteristic is composed of a set of related sub-characteristics. (ISO/IEC 25010)

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

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

Software Quality Characteristic—a category of software quality attributes that bears on software quality. (ISO/IEC 25010)

Software Quality Characteristic Measure—a software quality measure derived from measuring the attributes related to a specific software quality characteristic.

Software Quality Issue—architectural or coding practices that are known to cause problems in software development, maintenance, or operations and for which software quality rules can be defined that help avoid problems created by the issue.

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

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

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

Software Quality Property—measureable component of software quality. (derived from ISO/IEC 25010)

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

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

Software Performance Efficiency—the degree to which the performance of a piece of software minimizes its use of resources or time in accomplishing its specified functions under stated conditions. (adapted from ISO/IEC 25010)

Software Performance Efficiency Measure Element—a measure defined in terms of a quality attribute of software that affects its Performance Efficiency and the measurement method for quantifying it, including optionally the transformation by a mathematical function. (adapted from ISO/IEC 25023)

Structural Quality—the degree to which a set of static attributes of a software product satisfy stated and implied needs for the software product to be used under specified conditions—a component of software quality. This concept is referred to as internal software quality in ISO/IEC 25010.

Violation—a pattern or structure in the code that is inconsistent with good architectural and coding practices and can lead to problems in operation or maintenance.

5. Symbols (and Abbreviated Terms)

CISQ – Consortium for IT Software Quality

KDM – Knowledge Discovery Meta-model

SPMS – Structured Patterns Meta-model Standard

SMM – Structured Metrics Meta-model

6. Additional Information (Informative)

6.1 Software Product Inputs

The following inputs are needed by static code analyzers in order to interpret violations of the software quality rules that would be included in individual software quality measure elements.

- The entire source code for the application being analyzed
- All materials and information required to prepare the application for production
- A description of the architecture and layer boundaries of the application, including an assignment of modules to layers

Static code analyzers will also need a list of the violations that constitute each quality element in the Automated Source Code Performance Efficiency Measure.

6.2 Input Values for Thresholds in Measure Elements

Several of the weaknesses in the Automated Source Code Performance Efficiency measure detect violations of good architectural or coding practice based on threshold values for a construct being exceeded. Table 1 lists the default threshold value used in specifying this measure. In using this measure, threshold values can be adjusted to different levels. However, when the threshold values are adjusted the results cannot be compared or benchmarked to data from other analyses that used the default values. In such cases it may be good to compute values for both the default and adjusted values.

Table 1. Input Values for Thresholds in Measure Elements

<DataAccessComponentList> list of components designated to manage data accesses	
<NumberOfRowsThresholdValue> minimum value	The default value for <NumberOfRowsThresholdValue> is 1000000.
<NumberOfJoinsThresholdValue> maximum value <NumberOfSubQueriesThresholdValue> maximum value	The default value for <NumberOfJoinsThresholdValue> is 5. The default value for <NumberOfSubQueriesThresholdValue> is 3.

<IndexRangeThresholdValue> maximum value	The default value for <IndexRangeThresholdValue> is 10.
<NumberOfTableIndicesThresholdValue> maximum value	The default value for <NumberOfTableIndicesThresholdValue> is 3.
<NumberOfDataQueriesThresholdValue> maximal value	The default value for <NumberOfDataQueriesThresholdValue> is 2 client-side, and 5 server-side.
<NumberOfAggregatedObjectsThresholdValue> maximum value	The default value for <NumberOfAggregatedObjectsThresholdValue> is 5.

6.3 Automated Source Code Performance Efficiency Measure Elements

The violations of good architectural and coding practice incorporated into the Automated Source Code Performance Efficiency Measure are listed and describe in the following Table 2. Some of the CWEs from the Common Weakness Enumeration repository that are included in the CISQ Security measure are also defects that can cause Performance Efficiency problems. In order to retain consistency across measurement specifications, the original CWE numbers and titles have been retained for these Performance Efficiency measure elements.

Table 2. Performance Efficiency Patterns and Their Consequences, Objectives, and Measure Elements

Performance Efficiency Pattern	Consequence	Objective	Measure Element
ASCPEM-PRF-1: Static Block Element containing Class Instance Creation Control Element	Software that is coded so as to execute expensive computations repeatedly (such as in loops) requires excessive computational resources when the usage and data volume grow	Avoid upfront initialization of software data elements	Number of instances where a storable data element or member data element is initialized with a value in the 'Write' action and is located in a block of code which is declared as static
ASCPEM-PRF-2: Immutable Storable and Member Data Element Creation	Software featuring known under-efficient coding practices requires excessive computational resources	Avoid unnecessary usage of additional immutable data elements	Number of instances where a named callable control element or method control element creates immutable text data elements via the string concatenation statement (which could be avoided by using text buffer data elements)
ASCPEM-PRF-3: Static Member Data Element outside of	Software featuring known under-efficient coding	Avoid unnecessary up-front allocation	Number of instances where a static member element is declared as static but its parent

a Singleton Class Element	practices requires excessive computational resources	of memory for all data elements	class element is not a singleton class (that is, a class element that can be used only once in the 'to' association of a 'Create' action); it does not take into account final static fields
ASCPEM-PRF-4: Data Resource Read and Write Access Excessive Complexity	Software featuring known under-efficient SQL Query and Data Access constructs requires excessive computational resources	Avoid overly complex data queries	Number of instances where the number of rows in a data table exceeds a threshold value, and where it is accessed by a data action whose number of joins between tables exceeds a threshold value, and its number of sub-queries exceeds a threshold value. The default value for the number of rows is 1000000, the default value for the number of joins is 5, and the default value for the number of sub-queries is 3
ASCPEM-PRF-5: Data Resource Read Access Unsupported by Index Element	Software featuring known under-efficient SQL Query and Data Access constructs requires excessive computational resources	Avoid unnecessary full scans of data tables	Number of instances where the syntax of the 'ReadsColumnSet' action and the index configuration of an SQL table or SQL view causes the DBMS to run sequential searches
ASCPEM-PRF-6: Large Data Resource ColumnSet Excessive Number of Index Elements	Software featuring known under-efficient SQL Query and Data Access constructs requires excessive computational resources	Avoid too many indices on very large data tables	Number of instances where the number of rows in a data table exceeds a threshold value, and its number of indices exceeds a threshold value. The default value for number of rows is 1000000, and the default value for number of table indices is 3
ASCPEM-PRF-7: Large Data Resource ColumnSet with Index Element of Excessive Size	Software featuring known under-efficient SQL Query and Data Access constructs requires excessive computational resources	Avoid overly large indices on very large data tables	Number of instances where the number of rows in a data table exceeds a threshold value, and where the range value of its index exceeds a threshold value. The default value for number of rows is 1000000, and the default value for the index range is 10

ASCPEM-PRF-8: Control Elements Requiring Significant Resource Element within Control Flow Loop Block	Software that is coded so as to execute expensive computations repeatedly (such as in loops) requires excessive computational resources when the usage and data volume grow	Avoid resource consuming operations found directly or indirectly within loops	Number of instances where a control element that causes platform resource consumption is directly or indirectly called via an execution path starting from within a loop body block or within a loop condition
ASCPEM-PRF-9: Non-Stored SQL Callable Control Element with Excessive Number of Data Resource Access	Software that does not leverage database capabilities to efficiently run data processing (such as stored procedures and functions) requires excessive computational resources	Use dedicated stored procedures when multiple data accesses are needed	Number of instances where server-side non-stored callable control elements in a data manager resource embed a number of data resource accesses that exceed a threshold value. The default value for the number of data queries is 5
ASCPEM-PRF-10: Non-SQL Named Callable and Method Control Element with Excessive Number of Data Resource Access	Software that does not leverage database capabilities to efficiently run data processing (such as stored procedures and functions) requires excessive computational resources	Avoid software elements requiring too many data accesses outside of the data manager	Number of instances where a client-side control element named callable or method control element are not in any data manager resource and they embed a number of data resource access actions that exceed a threshold value. The default threshold for the number of data queries is 2.
ASCPEM-PRF-11: Data Access Control Element from Outside Designated Data Manager Component	Software deployed in distributed environment that does not maintain redundancy of data (such as cache) and code increases the time with which they are accessed	Use dedicated and specialized data manager component(s).	Number of instances where a named callable control element or method control element executes a data action that is not executed through a dedicated central data manager component identified in the data access component list (the unlisted data access component can be either client-side or server-side, which means that not all server-side components are allowed to handle data

			accesses and that data access components can be developed using non-SQL languages; in essence, the data access does not follow the intended design)
ASCPEM-PRF-12: Storable and Member Data Element Excessive Number of Aggregated Storable and Member Data Elements	Software featuring known under-efficient coding practices requires excessive computational resources	Avoid the creation of excessively large data elements	Number of instances where a data type of the storable data element aggregates a number of storable data elements with non-primitive data types that exceeds a threshold value. The default value for the number of aggregated objects is 5
ASCPEM-PRF-13: Data Resource Access not using Connection Pooling capability	Software featuring known under-efficient coding practices requires excessive computational resources	Share database connections via a connection pool	Number of instances where a named callable control element or method control element executes a data resource management action without using a connection pooling capability (the usage of connection pooling capability is technology dependent; for example, connection pooling is disabled with the addition of 'Pooling=false' to the connection string with ADO.NET or the value of 'com.sun.jndi.ldap.connect.pool' environment parameter in Java)
ASCPEM-PRF-14: Storable and Member Data Element Memory Allocation Missing De-Allocation Control Element	Software featuring known under-efficient coding practices requires excessive computational resources	Avoid failure to release used memory	Number of instances where a memory resource is explicitly allocated via the 'ManagesResource' action to a storable or member data element which is used throughout the application, and for which the transformation sequence is composed of action elements with data relations some of which are part of named callable and method control elements, but none of which is a memory release statement

ASCPEM-PRF-15: Storable and Member Data Element Reference Missing De-Referencing Control Element	Software featuring known under-efficient coding practices requires excessive computational resources	Avoid failure to release used data elements	Number of instances where a method control element references via the access action a storable or member data element without invoking its finalize method

7. SPMS Representation of the Performance Efficiency Quality Measure Patterns (Normative)

Introduction

This chapter displays in a human readable format the content of the machine readable XMI format file accompanying this specification. The content of the machine readable XMI format file is the representations of the CISQ Performance Efficiency Measure Elements:

- using the framework of the Structured Patterns Meta-model Standard (SPMS)
- describing the source code entities within a measure element as entities contained in the Knowledge Discovery Meta-model (KDM) and embedding them within the SPMS patterns, so that the representation is both descriptive and formal.

SPMS

More specifically, the machine readable XMI format file accompanying this specification uses the SPMS Definitions Classes:

- PatternDefinition (SPMS:PatternDefinition): the pattern specification. In the context of this document, each CISQ Quality Measure Element is basically the count of occurrences of the described patterns.
- Role (SPMS:Role): "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."
- PatternSection (SPMS:PatternSection): "A PatternSection is a free-form prose textual description of a portion of a PatternDefinition." In the context of this document, there are 6 different PatternSections in use:
    - "Descriptor" to provide pattern signature, a visible interface of the pattern,
    - "Measure Element" to provide a human readable explanation of the measure,
    - "Description" to provide a human readable explanation of the pattern that is sought after, identifying "Roles" and KDM modeling information,
    - "Objective" to provide a human readable explanation of the intent to get rid of the occurrences of the pattern that is sought after,
    - "Consequence" to provide a human readable explanation of the issue the detection of the pattern is designed to solve,
    - "Input" to provide a human readable of the parameters that are needed to fine-tune the behavior of the pattern detection (e.g.: the target application architectural blueprint to comply with)

- "Comment" to provide some additional information (until now, used to inform about situations where the same measure element is useful for another one of the categories)

As well as some of the SPMS Relationships Classes:

- MemberOf (SPMS:MemberOf): "An InterpatternRelationship specialized to indicate inclusion in a Category"
- Category (SPMS:Category): "A Category is a simple grouping element for gathering related PatternDefinitions into clusters." In the context of this document, the SPMS Categories are used to represent the Quality Characteristic of Performance Efficiency.

KDM

More specifically, the machine readable XMI format file accompanying this specification uses KDM entities in the "Description" section of the pattern definitions. Descriptions try to remain as generic yet as accurate as possible so that the pattern can be applicable and applied to as many situations as possible, such as different technologies and different programming languages. This means:

1. the descriptions include information such as (code:MethodUnit), (action:Reads), (platform:ManagesResource) etc. to identify the KDM entities in the pattern definition
2. the descriptions only elaborate the salient aspects of the pattern as the specifics can be technology- or language-dependent

Table 2 presents a translation of KDM terms used in the current specification into the wording of their ordinary usage.

Table 2. KDM Elements Incorporated into SPMS Patterns

Ordinary term(s)	KDM description(s)
function, method, procedure, stored procedure, subroutine, etc.	named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit)
variable, field, member, etc.	storable data element (code:StorableUnit) or member data element (code:MemberUnit)
class	class element (code:StorableUnit with code:DataType code:ClassUnit)
interface	interface element (code:StorableUnit of code:DataType code:InterfaceUnit)
method	method element (code:MethodUnit)
field, member	member element (code:MemberUnit)
SQL stored procedures	stored callable control elements (code:CallableUnit with code:CallableKind 'storec') in a data manager resource (platform:DataManager)

return code value	value (code:Value) of the return parameter (code:ParameterUnit of code:ParameterKind 'return')
exception	exception parameter (code:ParameterUnit with code:ParameterKind 'exception')
user input data flow	an external value is entered is entered into the application through the 'ReadsUI' user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the 'TransformationSequence' sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used as
execution path	execution path (action:BlockUnit composed of action:ActionElements with action:CallableRelations to code:ControlElements)
Libraries, etc.	deployed component (platform:DeployedComponent)
RDBMS	data manager resource (platform:DataManager)
loop body	loop body block (action:BlockUnit starting as the action:TrueFlow of the loop action:GuardedFlow and ending with an action:Flow back to the loop action:GuardedFlow)
loop condition	loop condition (action:BlockUnit used in the action:GuardedFlow)
singleton	class element (code:StorableUnit with code:DataType code:ClassUnit) that can be used only once in the 'to' association of a Create action (action:Creates)
checked	used by a check control element (code:ControlElement containing action:ActionElement with a kind from micro KDM list of comparison actions)

Reading guide

Sub-clause 7.2 represents the SPMS Category for the software quality characteristic covered in this specification. Starting with sub-clause 7.3, each section numbered 7.x represents a new SPMS PatternDefinition member of this SPMS Category. SPMS PatternDefinition sub-sections are:

- Pattern category: the “SPMS:Category” category that the pattern is related to through a “SPMS:MemberOf” relationship.
- Pattern sections: the list of "SPMS:PatternSection"sections from the pattern:
  - “Descriptor”,
  - “Description”,
  - “Objective”,

    - "Consequence",
    - and, when applicable,
        - "Input",
        - "Comment".
- Pattern roles: the list of "SPMS:Role" roles used in the "Descriptor" and "Description" sections above.

In the following sub-clauses:

- Data between square brackets (e.g.: [key CISQ Performance Efficiency]) identifies "xmi:id" that are unique and used to reference entities. They are machine-generated to ensure unicity.
- Data between paranthesis (e.g.: (code:MethodUnit)) identifies KDM modeling information.
- Data between angle brackets (e.g.: <ControlElement>) identifies SPMS Roles in Description and Input sections.

7.1. Category definition of CISQ Performance Efficiency

[key ASCPEM_Performance_Efficiency] CISQ Performance Efficiency

7.2. Pattern definition of ASCPEM-PRF-1: Static Block Element containing Class Instance Creation Control Element

Pattern Category

[key ASCPEM-PRF-1-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-1-objective]
Avoid upfront initialization of software data elements

Consequence

[key ASCPEM-PRF-1-consequence]
Software that is coded so as to execute expensive computations repeatedly (such as in loops) requires excessive computational resources when the usage and data volume grow

Measure Element

[key ASCPEM-PRF-1-measure-element]
Number of instances where a storable data element or member data element is initialized with a value in the 'Write' action and is located in a block of code which is declared as static

Description

[key ASCPEM-PRF-1-description]

This pattern identifies situations where a storable data element (code:StorableUnit) or member data element (code:MemberUnit) is initialized with a value in the <InitialisationStatement> Write action (action:Write) located into the <StaticBlock> block of code (action:BlockUnit) which is declared as static

Descriptor

[key ASCPEM-PRF-1-descriptor]
ASCPEM-PRF-1(StaticBlock: staticBlock,InitialisationStatement: initialisationStatement)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-1-roles-staticBlock] StaticBlock
[key ASCPEM-PRF-1-roles-initialisationStatement] InitialisationStatement

7.3. Pattern definition of ASCPEM-PRF-2: Immutable Storable and Member Data Element Creation

Pattern Category

[key ASCPEM-PRF-2-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-2-objective]
Avoid unnecessary usage of additional immutable data elements

Consequence

[key ASCPEM-PRF-2-consequence]
Software featuring known under-efficient coding practices requires excessive computational resources

Measure Element

[key ASCPEM-PRF-2-measure-element]
Number of instances where a named callable control element or method control element creates immutable text data elements via the string concatenation statement (which could be avoided by using text buffer data elements)

Description

[key ASCPEM-PRF-2-description]
This pattern identifies situations where the <ControlElement> named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit) creates immutable text data elements (code:DataElement with code:DataType

code:StringType) via the <StringConcatenationStatement> string concatenation statement (code:ControlElement using 2 action:Reads and 1 action:Writes on code:DataElement with code:DataType code:StringType), which could be avoided by using text buffer data elements.

Descriptor

[key ASCPEM-PRF-2-descriptor]
ASCPEM-PRF-2(ControlElement: controlElement,StringConcatenationStatement: stringConcatenationStatement)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-2-roles-controlElement] ControlElement
[key ASCPEM-PRF-2-roles-stringConcatenationStatement] StringConcatenationStatement

7.4. Pattern definition of ASCPEM-PRF-3: Static Member Data Element outside of a Singleton Class Element

Pattern Category

[key ASCPEM-PRF-3-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-3-objective]
Avoid unnecessary up-front allocation of memory for all data elements

Consequence

[key ASCPEM-PRF-3-consequence]
Software featuring known under-efficient coding practices requires excessive computational resources

Measure Element

[key ASCPEM-PRF-3-measure-element]
Number of instances where a static member element is declared as static but its parent class element is not a singleton class (that is, a class element that can be used only once in the 'to' association of a ‘Create’ action); it does not take into account final static fields

Description

[key ASCPEM-PRF-3-description]
This pattern identifies situations where the <StaticField> static member element (code:MemberUnit) is declared as static (code:StorableKind 'static') but its <ParentClass> parent class element (code:StorableUnit with code:DataType code:ClassUnit) is not a singleton class, that is, a class element

that can be used only once in the 'to' assoction of a Create action (action:Creates); it does not take into account final static fields (code:MemberUnit with code:StorableKind 'static' and code:ExportKind 'final').

Descriptor

[key ASCPEM-PRF-3-descriptor]
ASCPEM-PRF-3(StaticField: staticField,ParentClass: parentClass)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-3-roles-staticField] StaticField
[key ASCPEM-PRF-3-roles-parentClass] ParentClass

7.5. Pattern definition of ASCPEM-PRF-4: Data Resource Read and Write Access Excessive Complexity

Pattern Category

[key ASCPEM-PRF-4-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-4-objective]
Avoid overly complex data queries

Consequence

[key ASCPEM-PRF-4-consequence]
Software featuring known under-efficient SQL Query and Data Access constructs requires excessive computational resources

Measure Element

[key ASCPEM-PRF-4-measure-element]
Number of instances where the number of rows in a data table exceeds a threshold value, and where it is accessed by a data action whose number of joins between tables exceeds a threshold value, and its number of sub-queries exceeds a threshold value. The default value for the number of rows is 1000000, the default value for the number of joins is 5, and the default value for the number of sub-queries is 3

Description

[key ASCPEM-PRF-4-description]
This pattern identifies situations the <DataTable> data table (data:ColumnSet) is considered as very large, based on its <NumberOfRows> number of rows which exceeds the <NumberOfRowsThresholdValue> threshold value, and where it is accessed by the <QueryStatement>

data actions (data:DataActions) which is considered to be too complex, based on its <NumberOfJoins> number of joins between tables which exceeds the <NumberOfJoinsThresholdValue> threshold value, and its <NumberOfSubQueries> number of sub-queries which exceeds the <NumberOfSubQueriesThresholdValue> threshold value.
The default value for <NumberOfRowsThresholdValue> is 1000000.
The default value for <NumberOfJoinsThresholdValue> is 5.
The default value for <NumberOfSubQueriesThresholdValue> is 3.

Descriptor

[key ASCPEM-PRF-4-descriptor]
ASCPEM-PRF-4(DataTable: dataTable,NumberOfRows: numberOfRows, NumberOfRowsThresholdValue: numberOfRowsThresholdValue, QueryStatement: queryStatement, NumberOfJoins: numberOfJoins, NumberOfJoinsThresholdValue: numberOfJoinsThresholdValue, NumberOfSubQueries: numberOfSubQueries, NumberOfSubQueriesThresholdValue: numberOfSubQueriesThresholdValue)

Variable input

[key ASCPEM-PRF-4-input]
<NumberOfRowsThresholdValue> minimum value
<NumberOfJoinsThresholdValue> maximum value
<NumberOfSubQueriesThresholdValue> maximum value

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-4-roles-dataTable] DataTable
[key ASCPEM-PRF-4-roles-numberOfRows] NumberOfRows
[key ASCPEM-PRF-4-roles-numberOfRowsThresholdValue] NumberOfRowsThresholdValue
[key ASCPEM-PRF-4-roles-queryStatement] QueryStatement
[key ASCPEM-PRF-4-roles-numberOfJoins] NumberOfJoins
[key ASCPEM-PRF-4-roles-numberOfJoinsThresholdValue] NumberOfJoinsThresholdValue
[key ASCPEM-PRF-4-roles-numberOfSubQueries] NumberOfSubQueries
[key ASCPEM-PRF-4-roles-numberOfSubQueriesThresholdValue] NumberOfSubQueriesThresholdValue

7.6. Pattern definition of ASCPEM-PRF-5: Data Resource Read Access Unsupported by Index Element

Pattern Category

[key ASCPEM-PRF-5-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-5-objective]
Avoid unnecessary full scans of data tables

Consequence

[key ASCPEM-PRF-5-consequence]
Software featuring known under-efficient SQL Query and Data Access constructs requires excessive computational resources

Measure Element

[key ASCPEM-PRF-5-measure-element]
Number of instances where the syntax of the 'ReadsColumnSet' action and the index configuration of an SQL table or SQL view causes the DBMS to run sequential searches

Description

[key ASCPEM-PRF-5-description]
This pattern idendifies situations where the syntax of the <SelectSQLStatement> ReadsColumnSet action (data:ReadsColumnSet) and the index (data:Index) configuration of the <SQLTableOrView> SQL table (data:RelationalTable) or SQL view (data:RelationalView) causes the DBMS to run sequential searches.

Descriptor

[key ASCPEM-PRF-5-descriptor]
ASCPEM-PRF-5(SelectSQLStatement: selectSQLStatement,SQLTableOrView: sQLTableOrView)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-5-roles-selectSQLStatement] SelectSQLStatement
[key ASCPEM-PRF-5-roles-sQLTableOrView] SQLTableOrView

7.7. Pattern definition of ASCPEM-PRF-6: Large Data Resource ColumnSet Excessive Number of Index Elements

Pattern Category

[key ASCPEM-PRF-6-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-6-objective]
Avoid too many indices on very large data tables

Consequence

[key ASCPEM-PRF-6-consequence]
Software featuring known under-efficient SQL Query and Data Access constructs requires excessive computational resources

Measure Element

[key ASCPEM-PRF-6-measure-element]
Number of instances where the number of rows in a data table exceeds a threshold value, and its number of indices exceeds a threshold value. The default value for number of rows is 1000000, and the default value for number of table indices is 3

Description

[key ASCPEM-PRF-6-description]
This pattern identifies situations where the <DataTable> data table (data:ColumnSet) is considered as very large, based on its <NumberOfRows> number of rows which exceeds the <NumberOfRowsThresholdValue> threshold value, and is considered to have too many indices (data:Index), based on its <NumberOfTableIndices> number of indices which exceeds the <NumberOfTableIndicesThresholdValue> threshold value.
The default value for <NumberOfRowsThresholdValue> is 1000000.
The default value for <NumberOfTableIndicesThresholdValue> is 3.

Descriptor

[key ASCPEM-PRF-6-descriptor]
ASCPEM-PRF-6(DataTable: dataTable,NumberOfRows: numberOfRows, NumberOfRowsThresholdValue: numberOfRowsThresholdValue, NumberOfTableIndices: numberOfTableIndices, NumberOfTableIndicesThresholdValue: numberOfTableIndicesThresholdValue)

Variable input

[key ASCPEM-PRF-6-input]
<NumberOfRowsThresholdValue> minimum value
<NumberOfTableIndicesThresholdValue> maximum value

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-6-roles-dataTable] DataTable
[key ASCPEM-PRF-6-roles-numberOfRows] NumberOfRows
[key ASCPEM-PRF-6-roles-numberOfRowsThresholdValue] NumberOfRowsThresholdValue
[key ASCPEM-PRF-6-roles-numberOfTableIndices] NumberOfTableIndices
[key ASCPEM-PRF-6-roles-numberOfTableIndicesThresholdValue] NumberOfTableIndicesThresholdValue

7.8. Pattern definition of ASCPEM-PRF-7: Large Data Resource ColumnSet with Index Element of Excessive Size

Pattern Category

[key ASCPEM-PRF-7-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-7-objective]
Avoid overly large indices on very large data tables

Consequence

[key ASCPEM-PRF-7-consequence]
Software featuring known under-efficient SQL Query and Data Access constructs requires excessive computational resources

Measure Element

[key ASCPEM-PRF-7-measure-element]
Number of instances where the number of rows in a data table exceeds a threshold value, and where the range value of its index exceeds a threshold value. The default value for number of rows is 1000000, and the default value for the index range is 10

Description

[key ASCPEM-PRF-7-description]
This pattern identifies situations where the <DataTable> data table (data:ColumnSet) is considered as very large, based on its <NumberOfRows> number of rows which exceeds the <NumberOfRowsThresholdValue> threshold value, and where its <Index> index (data:Index) is considered as too large, based on its <IndexRange> range value which exceeds the <IndexRangeThresholdValue> threshold value.
The default value for <NumberOfRowsThresholdValue> is 1000000.
The default value for <IndexRangeThresholdValue> is 10.

Descriptor

[key ASCPEM-PRF-7-descriptor]
ASCPEM-PRF-7(DataTable: dataTable,NumberOfRows: numberOfRows, NumberOfRowsThresholdValue: numberOfRowsThresholdValue, Index: index, IndexRange: indexRange, IndexRangeThresholdValue: indexRangeThresholdValue)

Variable input

[key ASCPEM-PRF-7-input]
<NumberOfRowsThresholdValue> minimum value
<IndexRangeThresholdValue> maximum value

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-7-roles-dataTable] DataTable
[key ASCPEM-PRF-7-roles-numberOfRows] NumberOfRows
[key ASCPEM-PRF-7-roles-numberOfRowsThresholdValue] NumberOfRowsThresholdValue
[key ASCPEM-PRF-7-roles-index] Index
[key ASCPEM-PRF-7-roles-indexRange] IndexRange

[key ASCPEM-PRF-7-roles-indexRangeThresholdValue] IndexRangeThresholdValue

7.9. Pattern definition of ASCPEM-PRF-8: Control Elements Requiring Significant Resource Element within Control Flow Loop Block

Pattern Category

[key ASCPEM-PRF-8-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-8-objective]
Avoid resource consuming operations found directly or indirectly within loops

Consequence

[key ASCPEM-PRF-8-consequence]
Software that is coded so as to execute expensive computations repeatedly (such as in loops) requires excessive computational resources when the usage and data volume grow

Measure Element

[key ASCPEM-PRF-8-measure-element]
Number of instances where a control element that causes platform resource consumption is directly or indirectly called via an execution path starting from within a loop body block or within a loop condition

Description

[key ASCPEM-PRF-8-description]
This pattern identifies situations where the <ExpensiveControlElement> control element (code:ControlElement), whose nature is known to cause platform resource consumption (platform:PlatformActions with platform:ResourceType), is directly or indirectly called via the <ExecutionPath> execution path (action:BlockUnit composed of action:ActionElements with action:CallableRelations to code:ControlElements), starting from within the loop body block (action:BlockUnit starting as the action:TrueFlow of the loop action:GuardedFlow and ending with an action:Flow back to the loop action:GuardedFlow) or whithin the loop condition (action:BlockUnit used in the action:GuardedFlow)

Descriptor

[key ASCPEM-PRF-8-descriptor]
ASCPEM-PRF-8(LoopStatement: loopStatement,ExpensiveOperation: expensiveOperation, ExecutionPath: executionPath)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-8-roles-loopStatement] LoopStatement
[key ASCPEM-PRF-8-roles-expensiveOperation] ExpensiveOperation
[key ASCPEM-PRF-8-roles-executionPath] ExecutionPath

7.10. Pattern definition of ASCPEM-PRF-9: Non-Stored SQL Callable Control Element with Excessive Number of Data Resource Access

Pattern Category

[key ASCPEM-PRF-9-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-9-objective]
Use dedicated stored procedures when multiple data accesses are needed

Consequence

[key ASCPEM-PRF-9-consequence]
Software that does not leverage database capabilities to efficiently run data processing (such as stored procedures and functions) requires excessive computational resources

Measure Element

[key ASCPEM-PRF-9-measure-element]
Number of instances where server-side non-stored callable control elements in a data manager resource embed a number of data resource accesses that exceed a threshold value. The default value for the number of data queries is 5

Description

[key ASCPEM-PRF-9-description]
This pattern identifies situations where the server-side <ControlElement> non-stored callable control elements (code:CallableUnit without code:CallableKind 'stored') in the data manager resource (platform:DataManager), embeds <NumberOfDataQueries> number of data resource access (data:DataActions), which is considered as too large because it exceeds the <NumberOfDataQueriesThresholdValue> threshold value.
The default value for <NumberOfDataQueriesThresholdValue> is 5.

Descriptor

[key ASCPEM-PRF-9-descriptor]
ASCPEM-PRF-9(ControlElement: controlElement,NumberOfDataQueries: numberOfDataQueries, NumberOfDataQueriesThresholdValue: numberOfDataQueriesThresholdValue)

Variable input

[key ASCPEM-PRF-9-input]
<NumberOfDataQueriesThresholdValue> maximal value

Comment
(none applicable)

List of Roles
[key ASCPEM-PRF-9-roles-controlElement] ControlElement
[key ASCPEM-PRF-9-roles-numberOfDataQueries] NumberOfDataQueries
[key ASCPEM-PRF-9-roles-numberOfDataQueriesThresholdValue] NumberOfDataQueriesThresholdValue

7.11. Pattern definition of ASCPEM-PRF-10: Non-SQL Named Callable and Method Control Element with Excessive Number of Data Resource Access

Pattern Category
[key ASCPEM-PRF-10-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective
[key ASCPEM-PRF-10-objective]
Avoid software elements requiring too many data accesses outside of the data manager

Consequence
[key ASCPEM-PRF-10-consequence]
Software that does not leverage database capabilities to efficiently run data processing (such as stored procedures and functions) requires excessive computational resources

Measure Element
[key ASCPEM-PRF-10-measure-element]
Number of instances where a client-side control element named callable or method control element are not in any data manager resource and they embed a number of data resource access actions that exceed a threshold value. The default threshold for the number of data queries is 2.

Description
[key ASCPEM-PRF-10-description]
This pattern identifies situations where the client-side <ControlElement> named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), not in any data manager resource (platform:DataManager), embeds <NumberOfDataQueries> number of data resource access (data:DataActions), which is considered as too large because it exceeds the <NumberOfDataQueriesThresholdValue> threshold value.
The default value for <NumberOfDataQueriesThresholdValue> is 2.

Descriptor
[key ASCPEM-PRF-10-descriptor]
ASCPEM-PRF-10(ControlElement: controlElement,NumberOfDataQueries: numberOfDataQueries, NumberOfDataQueriesThresholdValue: numberOfDataQueriesThresholdValue)

Variable input

[key ASCPEM-PRF-10-input]
<NumberOfDataQueriesThresholdValue> maximal value

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-10-roles-controlElement] ControlElement
[key ASCPEM-PRF-10-roles-numberOfDataQueries] NumberOfDataQueries
[key ASCPEM-PRF-10-roles-numberOfDataQueriesThresholdValue]
NumberOfDataQueriesThresholdValue

7.12. Pattern definition of ASCPEM-PRF-11: Data Access Control Element from Outside Designated Data Manager Component

Pattern Category

[key ASCPEM-PRF-11-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-11-objective]
Use dedicated and specialized data manager component(s).

Consequence

[key ASCPEM-PRF-11-consequence]
Software deployed in distributed environment that does not maintain redundancy of data (such as cache) and code increases the time with which they are accessed

Measure Element

[key ASCPEM-PRF-11-measure-element]
Number of instances where a named callable control element or method control element executes a data action that is not executed through a dedicated central data manager component identified in the data access component list (the unlisted data access component can be either client-side or server-side, which means that not all server-side components are allowed to handle data accesses and that data access components can be developed using non-SQL languages; in essence, the data access does not follow the intended design)

Description

[key ASCPEM-PRF-11-description]
This pattern identifies situations where the <ControlElement> named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit) executes the <SQLStatement> data action (data:DataActions including data:ReadsColumnSet or data:WritesColumnSet) although it is not part of the <CentralDataManager> component (structure:Component) identified as one of the dedicated data access component from the

<DataAccessComponentList> list. the <Component> component can be either client-side either server-side, which means that not all server-side components are allowed to handle data accesses. The data access component can be either client-side either server-side, which means that data access components can be developped using non-SQL languages. The pattern simply identifies situations where the implementation does not follow the intended design, regardless of the design.

Descriptor

[key ASCPEM-PRF-11-descriptor]
ASCPEM-PRF-11(ControlElement: controlElement,SQLStatement: sQLStatement, DataAccessComponentList: dataAccessComponentList)

Variable input

[key ASCPEM-PRF-11-input]
<DataAccessComponentList> list of components designated to manage data accesses

Comment

[key ASCPEM-PRF-11-comment] Measure element contributes to Performance Efficiency and Reliability (as RLB-10)

List of Roles

[key ASCPEM-PRF-11-roles-controlElement] ControlElement
[key ASCPEM-PRF-11-roles-sQLStatement] SQLStatement
[key ASCPEM-PRF-11-roles-dataAccessComponentList] DataAccessComponentList

7.13. Pattern definition of ASCPEM-PRF-12: Storable and Member Data Element Excessive Number of Aggregated Storable and Member Data Elements

Pattern Category

[key ASCPEM-PRF-12-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-12-objective]
Avoid the creation of excessively large data elements

Consequence

[key ASCPEM-PRF-12-consequence]
Software featuring known under-efficient coding practices requires excessive computational resources

Measure Element

[key ASCPEM-PRF-12-measure-element]
Number of instances where a data type of the storable data element aggregates a number of storable data elements with non-primitive data types that exceeds a threshold value. The default value for the number of aggregated objects is 5

Description

[key ASCPEM-PRF-12-description]
This pattern identifies situations where the data type (code:DataType) of the <AggregatingDataElement> storable data element (code:StorableUnit) aggregates <NumberOfAggregatedDataElements> storable data elements with non-primitive data types (code:DataType from code:PrimitiveType), which is considered as too large because it exceeds the <NumberOfAggregatedObjectsThresholdValue> threshold value.
The default value for <NumberOfAggregatedObjectsThresholdValue> is 5.

Descriptor

[key ASCPEM-PRF-12-descriptor]
ASCPEM-PRF-12(AggregatingDataElement: aggregatingDataElement,NumberOfAggregatedDataElements: numberOfAggregatedDataElements, NumberOfAggregatedObjectsThresholdValue: numberOfAggregatedObjectsThresholdValue)

Variable input

[key ASCPEM-PRF-12-input]
<NumberOfAggregatedObjectsThresholdValue> maximum value

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-12-roles-aggregatingDataElement] AggregatingDataElement
[key ASCPEM-PRF-12-roles-numberOfAggregatedDataElements] NumberOfAggregatedDataElements
[key ASCPEM-PRF-12-roles-numberOfAggregatedObjectsThresholdValue] NumberOfAggregatedObjectsThresholdValue

7.14. Pattern definition of ASCPEM-PRF-13: Data Resource Access not using Connection Pooling capability

Pattern Category

[key ASCPEM-PRF-13-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-13-objective]
Share database connections via a connection pool

Consequence

[key ASCPEM-PRF-13-consequence]
Software featuring known under-efficient coding practices requires excessive computational resources

Measure Element

[key ASCPEM-PRF-13-measure-element]

Number of instances where a named callable control element or method control element executes a data resource management action without using a connection pooling capability (the usage of connection pooling capability is technology dependent; for example, connection pooling is disabled with the addition of 'Pooling=false' to the connection string with ADO.NET or the value of 'com.sun.jndi.ldap.connect.pool' environment parameter in Java)

Description

[key ASCPEM-PRF-13-description]
This pattern identifies situations where the <ControlElement> named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit) executes the <SQLConnectionInitializationStatement> data resource management action (platform:ManagesResource with platform:DataManager) not using connection pooling capability.
The usage of connection pooling capability is technology dependent. As examples, connection pooling is disabled with the addition of 'Pooling=false' to the connection string with ADO.NET and the value of 'com.sun.jndi.ldap.connect.pool' environment parameter in Java.

Descriptor

[key ASCPEM-PRF-13-descriptor]
ASCPEM-PRF-13(ControlElement: controlElement,SQLConnectionInitializationStatement: sQLConnectionInitializationStatement)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-13-roles-controlElement] ControlElement
[key ASCPEM-PRF-13-roles-sQLConnectionInitializationStatement] SQLConnectionInitializationStatement

7.15. Pattern definition of ASCPEM-PRF-14: Storable and Member Data Element Memory Allocation Missing De-Allocation Control Element

Pattern Category

[key ASCPEM-PRF-14-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-14-objective]
Avoid failure to release used memory

Consequence

[key ASCPEM-PRF-14-consequence]

Software featuring known under-efficient coding practices requires excessive computational resources

Measure Element

[key ASCPEM-PRF-14-measure-element]
Number of instances where a memory resource is explicitly allocated via the 'ManagesResource' action to a storable or member data element which is used throughout the application, and for which the transformation sequence is composed of action elements with data relations some of which are part of named callable and method control elements, but none of which is a memory release statement

Description

[key ASCPEM-PRF-14-description]
This pattern identifies situations where a memory resource (platform:RuntimeResource) is explicitelly allocated via the <MemoryAllocationStatement> ManagesResource action (platform:ManagesResources) to the <DataElement> storable or member data element (code:StorableUnit or code:MemberUnit), which is used throughout the application, along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), none of which being a memory release statement (platform:ManagesResource).

Descriptor

[key ASCPEM-PRF-14-descriptor]
ASCPEM-PRF-14(MemoryAllocationStatement: memoryAllocationStatement,TransformationSequence: transformationSequence)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-14-roles-memoryAllocationStatement] MemoryAllocationStatement
[key ASCPEM-PRF-14-roles-transformationSequence] TransformationSequence

7.16. Pattern definition of ASCPEM-PRF-15: Storable and Member Data Element Reference Missing De-Referencing Control Element

Pattern Category

[key ASCPEM-PRF-15-relatedPatts-Performance Efficiency] ASCPEM_Performance_Efficiency

Pattern Sections

Objective

[key ASCPEM-PRF-15-objective]

Avoid failure to release used data elements

Consequence

[key ASCPEM-PRF-15-consequence]
Software featuring known under-efficient coding practices requires excessive computational resources

Measure Element

[key ASCPEM-PRF-15-measure-element]
Number of instances where a method control element references via the access action a storable or member data element without invoking its finalize method

Description

[key ASCPEM-PRF-15-description]
This pattern identifies situations where the <MethodElement> method control elements (code:MethodUnit) references via the <ReferenceStatement> access action (action:DataRelations) the <ReferencedObject> storable or member data element (code:StorableUnit or code:MemberUnit) without invoking its finalize method (code:MethodUnit with code:MethodKind 'destructor')

Descriptor

[key ASCPEM-PRF-15-descriptor]
ASCPEM-PRF-15(MethodElement: methodElement,ReferenceStatement: referenceStatement, ReferencedObject: referencedObject)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCPEM-PRF-15-roles-methodElement] MethodElement
[key ASCPEM-PRF-15-roles-referenceStatement] ReferenceStatement
[key ASCPEM-PRF-15-roles-referencedObject] ReferencedObject

8. Calculation of the Automated Source Code Performance Efficiency Measure and Functional Density (Normative)

8.1 Calculation of the Base Measure

A count of total violations of quality rules was selected as the best alternative for measurement. Software quality measures have frequently been scored at the component level and then aggregated to develop an overall score for the application. However, scoring at the component level was rejected because many critical violations of Performance Efficiency quality rules cannot be isolated to a single component, but rather involve interactions among several components. Therefore, the Automated Source Code Performance Efficiency Measure is computed as the sum of its 15 quality measure elements computed across the entire application.

The calculation of the Automated Source Code Performance Efficiency Measure begins with determining the value of each of the 15 Performance Efficiency measure elements. Each Performance Efficiency measure element is measured as the total number of violations of its associated quality rule that are detected through automated analysis. Thus the value of each of the 15 Performance Efficiency measure elements is represented as CISQ-PrfME$_i$ where the range for i runs from 1 to 15.

$$\text{CISQ-PrfME}_i = \sum \text{(all violations of type CISQ-PrfME}_i \text{ detected through automated analysis)}$$

The value of the un-weighted and un-normalized Automated Source Code Performance Efficiency Measure (CISQ-Prf) is the sum of the values of the 15 Performance Efficiency measure elements.

$$\text{CISQ-Prf} = \sum_{i=1}^{15} \text{CISQ-PrfME}_i$$

Higher values of CISQ-Prf indicate a larger number of Performance Efficiency-related defects in the application.

8.2 Functional Density of Performance Efficiency Violations

In order to better compare Performance Efficiency results among different applications, the Automated Source Code Performance Efficiency Measure can be normalized by size to create a density measure. There are several size measures with which the density of Performance Efficiency violations can be normalized, such as lines of code, and function points. These size measures, if properly standardized, can be used for creating a density measure for use in benchmarking applications. However, the OMG Automated Function Points measure offers an automatable size measure that, as an OMG Supported Specification, is standardized, adapted from the International Function Point User Group's (IFPUG) counting guidelines, and commercially supported. Although other size measures can be legitimately used to evaluate the density of Performance Efficiency violations, the following density measure for Performance Efficiency violations is derived from OMG supported specifications for Automated Function Points and the Automated Source Code Performance Efficiency Measure. Thus, the functional density of CISQ Performance Efficiency violations is a simple division expressed as follows.

$$\text{CISQ-Prf-density} = \frac{\text{CISQ-Prf}}{\text{AFP}}$$

9. Alternative Weighted Measures and Uses (Informative)

9.1 Additional Derived Measures

There are many additional weighting schemes that can be applied to the Automated Source Code Performance Efficiency Measure or to the Performance Efficiency measure elements that compose it. Table 3 presents several candidate weighted measures and their potential uses. However, these weighting schemes are not derived from any existing standards and are therefore not normative.

Table 3. Informative Weighting Schemes for Performance Efficiency Measurement

Weighting scheme	Potential uses
Weight each Performance Efficiency measure by its severity	Measuring risk of performance problems such as degraded response time or excessive resource use
Weight each Performance Efficiency measure element by its effort to fix	Measuring cost of ownership, estimating future corrective maintenance effort and costs
Weight each module or application component by its density of Performance Efficiency violations	Prioritizing modules or application components for corrective maintenance or replacement

10. References (Informative)

Consortium for IT Software Quality (2010). http://www.it-cisq.org

Curtis, B. (1980). Measurement and experimentation in software engineering. *Proceedings of the IEEE*, 68 (9), 1103-1119.

International Organization for Standards (2012). ISO/IEC 25010 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models

International Organization for Standards (2012). *ISO/IEC 25023 (in development) Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality*.

International Organization for Standards (2003). *ISO/IEC TR 9126-3:2003, Software engineering — Product quality — Part 3: Internal metrics.*

Object Management Group (2014). Automated Function Points. formal 2014-01-03 http://www.omg.org/spec/AFP/, .

Appendix A: 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. Figure 1 displays the ISO/IEC 25010 software product quality model with the four software quality characteristics selected for automation by CISQ highlighted in orange. Each software quality characteristic is shown with the sub-characteristics that compose it.