Date: December 4, 2015

Automated Enhancement Points

Request for Comment

OMG Document Number: **admtf/2015-12-01**

Standard document URL: **http://www.omg.org/spec/AEP/1.0**

Associated files: Normative:
http://www.omg.org/spec/AEP/20151204/AutomatedEnhancementPoints.xmi

---

Submitted by:

CAST Software

Supporting organizations:

Accenture
Huawei

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 its 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

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.

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

0. Submission-Specific Material

0.1 Submission Preface

This submission is of a metric represented in compliance with OMG's Structured Metrics Meta-Model (SMM), Knowledge Discovery Meta-model (KDM, and Object Constraint Language (OCL). However, its submission is independent of these meta-models in order to establish it as a supported specification in its own right. The measure described in this specification is an important component for achieving the mission of the Architecture Driven Modernization Task Force and Consortium for IT Software Quality.

0.2 Copyright Waiver

CAST Software, Accenture, and Huawei: (i) grants 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, and (ii) grants to each member of the OMG a nonexclusive, royalty-free, paid up, worldwide license to make up to fifty (50) copies of this document for internal review purposes only and not for distribution, and (iii) 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 any OMG specification that may be based hereon or having conformed any computer software to such specification.

IPR Mode: Non-Assertion Covenant

0.3 Submitter Representative

Bill Curtis
CAST Software, Inc.
b.curtis@castsoftware.com

0.4 Supporting Organizations

Huawei
Accenture

0.5 Author Team

Philippe-Emmanuel Douziech
CAST Software, Inc.
p.douziech@castsoftware.com

Philippe Guerin
CAST Software, Inc.
p.guerin@castsoftware.com

Bill Curtis
Cast Software, Inc.
b.curtis@castsoftware.com

Jiuxing Wang
Huawei
wangjiuxing@huawei com

Jun Bu
Huawei
bujun@huawei.com

Amol Kumar
Accenture
amol.kumar@accenture.com

K. Srinivasan Rao
Accenture
k.srinivasa.rao@accenture.com

Deepa Jayakumar
Accenture
deepa.jayakumar@accenture.com

0.6 Proof of Concept

Automated Function Point analysis and measurement software based on the analysis of source code are offered commercially by CAST Software, and others. Automated Function Points has been adopted by numerous companies and the French government as a standard size measure for software. This specification for Automated Enhancement Points extends the coverage of automated functional measurement to use in measuring the size of software work performed during maintenance and enhancement projects.

1. Scope

1.1 Purpose

The purpose of this specification is to establish an automated sizing measure, Automated Enhancement Points, which solves specific problems with OMG's approved specification for Automated Function Points when used during maintenance and enhancement work. Current functional sizing measures frequently produce inaccurate and misleading results when comparing size to the effort expended during maintenance and enhancement. This problem is especially acute for analyzing productivity or evaluating contract performance. The solution proposed in Automated Enhancement Points establishes a standard measure that addresses 1) the complexity of the requested change, 2) anomalies in counting practices, and 3) measurement of the non-functional elements of an application. Automated Enhancement Points extend beyond the user transactional functions of a business application, the domain of functional measurement, to incorporate the components and elements that manage the non-functional, structural requirements of the application and allow it to perform in a modern multi-technology, multi-platform environment. As an additional benefit, this expansion beyond functional elements provides a foundation for extending automated software size measurement to OMG initiatives such as the Internet of Things Consortium that involve highly algorithmic or embedded software, a domain that traditionally measured size only in lines of code.

1.2 Problems in Sizing Maintenance and Enhancement Work

There are three primary problems when measuring the size of work completed during maintenance and enhancement activities with functional measures such as Automated Function Points. The first problem involves differences in complexity of satisfying a change request. The second problem involves a counting anomaly in software sizing analysis where some activities to go unmeasured and others cancel out size gains with deletions. The third problem involves the failure to account for work performed on code that is not addressed by either IPFUG counting practice guidelines (ISO/IEC, 2009) or Automated Function Points (OMG, 2014). All three of these issues can cause excessive and unaccounted for variation in productivity analyses. Without correcting for this variation, productivity analyses can be misleading and effort estimates can have unacceptably large margins of error.

1.2.1 Problem 1 — Complexity of Changes

Requests to make changes involving enhancements, deletions, or modifications to an application can vary widely in difficulty although they affect the same number of Automated Function Points. The difficulty involved in making a change can reside in either the complexity of the change or the complexity of the code to which the change is being made, or both. Some changes can be local and limited to a single object or file, while others may require changes or have impacts propagated across several objects or files. The effort involved in these changes is quite different because of the need to examine possible unintended side effects from more complex changes and propagated effects. When the number of Automated Function Points does not differ between two different levels of change complexity, the effort required to the more complex change can be severely underrepresented.

1.2.2 Problem 2 — Counting Anomalies

Maintenance and enhancement activities can involve the addition, modification, or deletion of code within a release. When all of the affected code is measured within the functional elements of an application, the functional size of added code can be partially or fully offset by the functional size of deleted code between two releases. For instance, current counting practices can cause the addition of 50 Automated Function Points to the code to be counted as a contribution of only 5 Automated Function Points if 45 Automated Function Points of dead or unneeded code are deleted from the application. Although substantial effort may have been expended on these maintenance and enhancement activities, the final functional size measure for the change between two releases is small or even zero, implying that little to no work was performed since the previous release. In addition, modifying code may not change the number of manually counted or Automated Function Points if it changes the attributes of, but not the number of functional elements counted.

As a result of these counting anomalies, the amount of work performed is frequently underrepresented by IFPUG counting guidelines or Automated Function Point counters. Productivity, effort, or cost analyses using size measures with these counting anomalies can contain wide variations in results that are caused more from unrepresentative calculations than from actual performance. Consequently, productivity analyses, calibration of cost estimates, benchmarking, and other analyses that compare size measures between releases are inaccurate.

1.2.3 Problem 3 — Unaddressed Structural Components

Automated Function Points and the International Function Point Users Group (IFPUG) counting guidelines from which they were derived only address the user transaction components of a business application, leaving as much as half the application unaddressed in when measuring size. Both automated and manual Function Points measure size as it relates to elements of the data flow and storage that support the user transactions within an application. The software unaddressed by traditional functional size measures are designated as non-functional because they must be performed by the software in order to support the completion of a user transaction. In essence, functional components allow users to perform their transactions, while the non-functional, structural components address the technical requirements that allow user transactions to be performed. Although functional requirements are usually platform-independent, functional requirements that primarily reside in the structural aspects of a system are dependent on attributes of the application's architecture, technology, and platform environment.

Examples of non-functional, structural components and elements of software include, but are not limited to:

- Validating data entry
- Mathematical operations
- Embedded/real-time response
- Formatting data
- Managing interactions between technologies or platforms
- Providing help

The problems described in clauses 1.2.2 and 1.2.3 are summarized visually in Figure 1.1, with a focus in Figure 1.2 on problem from clause 1.2.2. The problem describe in clause 1.2.1 is summarized visually in Figure 1.3, where LOC = Lines Of Code.

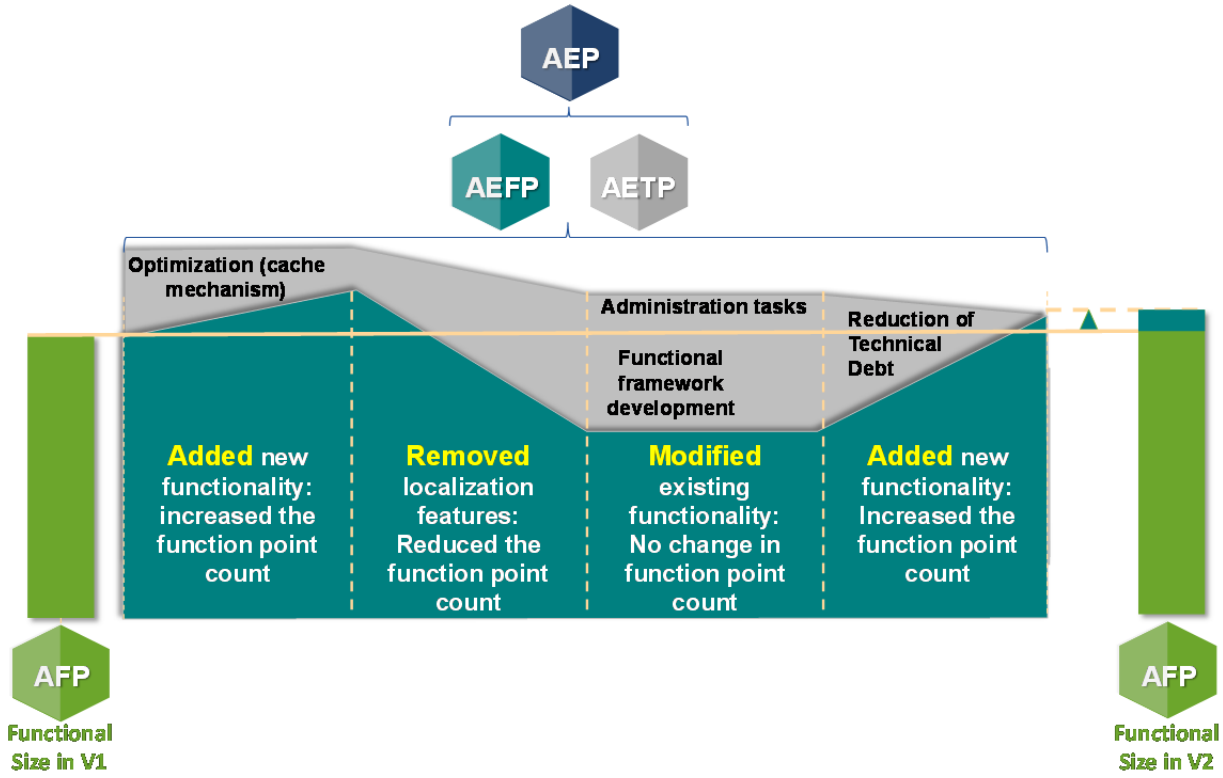

Figure 1.1. Problems in using Automated Function Points to size maintenance and enhancement work.

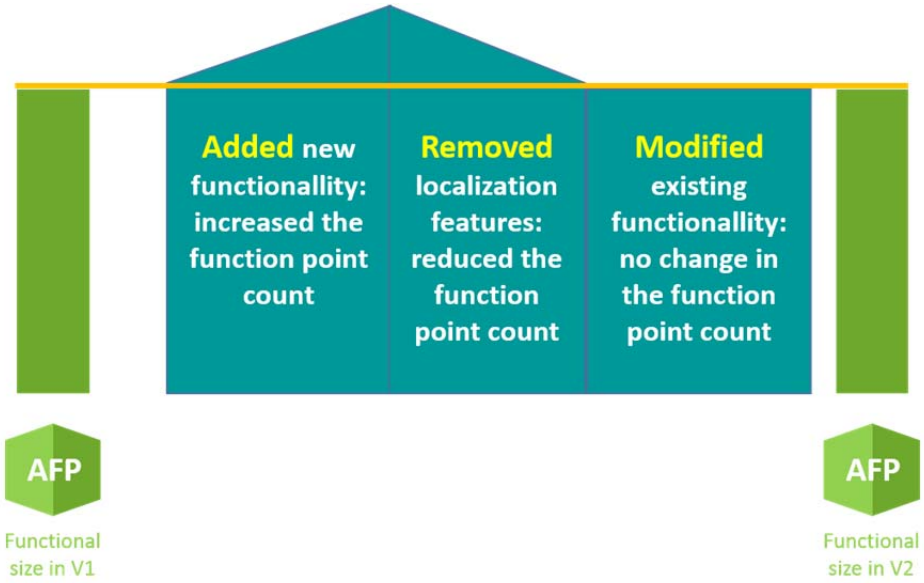

Figure 1.2. Counting anomalies in using Automated Function Points to size maintenance and enhancement work – full offset of added function points by deleted function points.

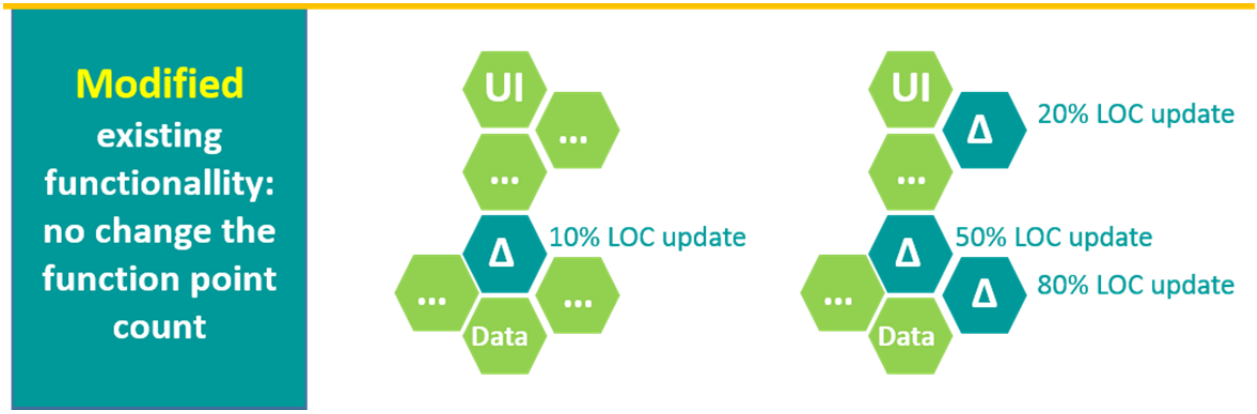

Figure 1.3. Complexity of changes issue when using Automated Function Points to size maintenance and enhancement.

1.3 Limitations of Existing Solutions to the Enhancement Sizing Problem

Solutions have been proposed for providing a function point-based measure to solve maintenance and enhancement sizing problems have been proposed by NESMA and IFPUG. The Netherlands Software Measurement Users Association (NESMA) proposed a method for solving the first two issues in clause 1.2, that is, removing the counting anomalies and adjusting for the complexity of the change. Their *Function Point Analysis for Software Enhancement* (NESMA, 2009) is a method for evaluating change requests and enhancement proposals to estimate the number of Function Points to be credited for maintenance and enhancement work based on data evolutions. Although Automated Enhancement Points will build on some of NESMA's concepts, the NESMA method does not specify how to count the non-functional elements of an application, nor does it provide a specification to deal with algorithmic non-data-related evolutions.

The International Function Point Users Group (IFPUG) is developing a method for measuring the non-functional attributes of an application and its environment called the *Software Non-functional Assessment Proc*ess (SNAP). SNAP measures assess 14 non-functional elements in 4 categories as shown in Table 1. Computational elements within an application that perform these non-functional actions are not sized by traditional Function Point measures. Consequently SNAP refers to these non-functional elements as the Technical components of an application. The specification of Automated Enhancement Points will retain 'Technical' in referring to the non-functional components of an application and their measures.

Data Operations	Interface Design	Technical Environment	Architecture
Data entry	User interface changes	Multiple platforms	Component-based software
Logical/math operations	Help methods	Database technology	Multiple I/O interfaces
Data formatting	Multiple input methods	Batch processes	
Internal data movements	Multiple output methods		
Delivering added value to users by data configuration			

Table 1. Non-functional (Technical) categories and elements in IFPUG's SNAP method

SNAP is defined primarily for sizing an application from its requirements and is not defined in a manner that allows them to be calculated from source code. Consequently SNAP relies on requirements analysis to assign the right categories and sub-categories to non-functional elements in an application. Since this determination cannot be achieved through automated analysis of the source code, the Automated Enhancement Points specification relies on analyzing changes in the software source code, regardless of the SNAP categories to which they would be assigned. Since SNAP measures are separate from Function Point counts within the IFPUG guidelines, no single size measure is calculated that incorporates both. Finally, SNAP measures provide a sizing specification that could be automated only for data element evolution. Therefore, while SNAP provides conceptual input, it is not sufficiently detailed for automating the sizing of work performed during a revision or release.

1.4 Development of the Automated Enhancement Points Measure

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 size and quality attributes from source code and submit them to OMG for approval as standards. Once approved these measures can be used by IT organizations, IT service providers, and software vendors in contracting, developing, testing, and accepting software applications. The Automated Function Points specification was approved as an OMG standard in 2014 and was quickly adopted by numerous public and private IT organizations.

One frequent use case motivating IT organizations to deploy Automated Function Points was for evaluating the productivity of maintenance and enhancement tasks. However, they often found large variations in their results that were difficult to interpret. Small changes often produced large Automated Function Point counts, while large changes often produced very few. Since they were unable to interpret productivity information reliably, CISQ sponsors prioritized this problem for solution. From Q4 2014 through Q3 2015 CISQ sponsors developed the concept and specification for Automated Enhancement Points. Two sponsors ran scripts on test applications to evaluate the measure's performance. The following specification represents a consensus among CISQ sponsors on a specification for Automated Enhancement Points. The name Enhancement Points was chosen because the objective of the measure is to size maintenance and enhancement work with a measure that sizes more than the functional attributes of an application.

2. Conformance

Implementations of this specification should be able to demonstrate all four of 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 and heuristics it uses in sufficient detail that the calculations may be independently verified by third parties. Clause 7.21 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. Normative References

The following OMG 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.

- 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
- Object Constraint Language, version 2.4 (OCL), formal/2014-02-03
- Automated Function Points (AFP), formal/2014-01-03
- ISO/IEC 20926:2009 — Software measurement — IFPUG Functional Size Measurement Method 2009.

4. Terms and Definitions

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

Application Model

composed of the computational objects in the source code and their relationships, some of which may contain processing rules and logic.

Application Scope

all computational objects within the boundary of a software application.

Artifact

a computational object in a software application that is callable by name to perform some processing that can be either functional or technical. *(*KDM's code:MethodUnit, code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored', with non-empty 'name' attribute; *see Appendix A for examples in mainstream technologies)*

Artifact Effort Complexity

assesses the complexity of implementing the Artifact or changes to it based a composite score of five software metrics that assess the complexity of the software environment in which the Artifact is embedded: McCabe Cyclomatic Complexity, Lines of Code, Lines of Comment Code, Fan-In, and SQL Complexity

Automated Enhancement Function Points (AEFP)

an automated measure for sizing changes made to computational objects in the Functional Output between two releases or revisions of an application. *(adpted from NESMA, 2009)*

Automated Enhancement Function Points Scope, a.k.a. Functional Output

All computational objects within the boundary of a software application that compose the modified functional features available in evolved user transactions and are measured by Automated Function Points.

Automated Enhancement Points (AEP)

an automated measure for sizing changes made to computational objects in the Application Scope between two releases or revisions of an application.

Automated Enhancement Technical Points (AETP)

an automated measure for sizing the changes made to computational objects in the Technical Output between two releases.

Automated Enhancement Technical Points Scope, a.k.a. Technical Output

all computational objects within the Automated Technical Points Scope that are added, modified, or deleted between two releases.

Automated Function Points (AFP)

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)*

Automated Function Points Data Entity

data element within the Automated Function Point Scope that supports the implementation of data entities. *(Automated Function Points, formal 2014-01-03)*

Automated Function Points Equivalent (AFP_{eq})

an adjustment of Automated Technical Points or Automated Enhancement Technical Points using the Equivalence Ratio to convert Implementation Points calculated for the Automated Enhancement Technical Points Scope into a statistically equivalent value of Automated Enhancement Function Points.

Automated Function Points Implementation Artifacts

all artifacts within the Automated Functional Points Implementation Scope.

Automated Function Points Implementation Scope

all computational objects within the boundary of a software application that compose the functional features available and are measured by Automated Function Points .

Automated Function Points Transaction Entry Point

computational object that support the interface with end users or third party applications, and used in the Automated Function Points specification as starting points of code paths towards data entities. *(formal 2014-01-03)*

Automated Function Points Transaction Implementation Scope

all computational objects in the Automated Function Points Implementation Scope that are within the boundary of a single transaction and are measured by Automated Function Points.

Automated Technical Points (ATP)

a specification for automating the counting and sizing the non-functional, structural (Technical) elements of an application (i.e., all computational objects not included in the Automated Function Points Implementation Scope) in a manner similar to that used in calculating Automated Function Points.

Automated Technical Points Implementation Artifacts

all artifacts within the Automated Technical Points Scope.

Automated Technical Points Implementation Scope

all Artifacts within the boundary of a software application that create and support the software technical foundation but are not measured by Automated Function Points, that is, Automated Technical Points Scope is the non-overlapping complement of Automated Function Points Scope with respect to the whole Application.

Boundary

a conceptual interface between an ensemble of computational objects and entities external to the boundary with which they interact; thus boundaries can be defined at many levels such as the whole application, any rigorously defined scope within the application, transactions, etc.

Complexity Factor (CF)

a value used in calculating Automated Enhancement Function Points for transactions that adjusts for

different complexities between additions, modifications, or deletions, and as well as the contributions of shared objects.

Computational Object

a code element that can be detected during static analysis (KDM's code:ComputationalObject).

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)*

Effort

the measurement of an organization's investment in a project or defined collection of tasks that is measured in person hours, person days, or other locally defined unit of human work that is consistent in the work categories, staff positions, and time periods to be included in the measure.

Effort Complexity (EC)

assesses the complexity of adding, modifying, or deleting an Artifact based a composite score of five software metrics that assess the complexity of the software environment in which the Artifact is embedded, that is, its size, comment level, algorithmic complexity, data access complexity, and coupling.

Enhancement

A change involving an addition, modification, or deletion applied to the software base of instructions comprising an application.

Equivalence Ratio (ER)

Application-specific and release-specific conversion ratio, for converting size measured in Implementation Points to a measure that is statistically equivalent to Automated Function Points.

Function Points (FP)

a measure of software size calculated according to the counting practices of the International Function Points User Group guidelines. *(IFPUG, 2014)*

Implementation Complexity

can be measured at several levels in Implementation Points by the sum of the Effort Complexities of all Artifacts within the scope boundary selected.

Implementation Points (IP)

a software sizing measure that accounts for the difficulty of implementing or changing software Artifacts by aggregating the Effort Complexity scores for Artifacts within the scope of the measure to which Implementation Points are applied.

Software Measure Element

a measure defined in terms of an attribute of software that affects size, and the measurement method for quantifying it, including optionally the transformation by a mathematical function. *(adapted from ISO/IEC 25023)*

Software Product

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

Technical Scope of Measurement

all software measure elements that are measured as part of calculating Automated Technical Points or Automated Enhancement Technical Points or any of their derivative measures.

5. Symbols (and Abbreviated Terms)

AEFP – Automated Enhanced Function Points

AEP – Automated Enhancement Points

AER – Adjusted Equivalence Ratio

AETP – Automated Enhancement Technical Points

AFP – Automated Function Points

ATP – Automated Technical Points

CF – Complexity Factor

CISQ – Consortium for IT Software Quality

EC – Effort Complexity

ER – Equivalence Ratio

FP – Function Points

AFP_{eq} – Automated Function Points equivalent

IP – Implementation Points

KDM – Knowledge Discovery Meta-model

OCL – Object Constraint Language

OMG – Object Management Group

SMM – Structured Metrics Meta-model

6. Method for Calculating Automated Enhancement Points (Informative)

6.1 Overview of Automated Enhancement Points

6.1.1 Application Level Scopes and Measures

The typical work performed by maintenance and development teams includes activities to support both the functioning of software features visible to end-users through their transactions (functional software), and the functioning of the software itself, as an autonomous entity that runs in a specific environment (technical software). To solve the problem of unaddressed technical components, scopes of measurement must be defined for both the computational elements, both functional and technical, of an application. The computational elements containing the processing logic will be referred to as **Artifacts** throughout the remainder of this specification, and are the types of elements that can be described using the Knowledge Discovery Meta-model (KDM's code:MethodUnit, code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored', with non-empty 'name' attribute.). Appendix A contains a list of Data Types that are considered Artifacts in programming languages commonly used in IT systems. Thus, the Artifacts within the boundary of an application will be distributed among the following scopes:

- **Application Scope**—all Artifacts within the boundary of a software application.
- **Automated Function Point Scope**—all computational elements within the boundary of a software application that compose the functional features available in user transactions and are measured by Automated Function Points.
- **Automated Technical Point Scope**—all computational elements within the boundary of a software application that are not measured by Automated Function Points, that is, Artifacts which construct the technical foundation that enables the execution of the application's functional features.

Within these scopes we can define two size measures:

- **Automated Function Points (AFP)**—the measure specified in OMG's approved specification *formal/2014-01-03.*
- **Automated Technical Points (ATP)**—a sizing measure for the Artifacts of an application not measured by Automated Function Points. Although this measure is outside the scope of this specification, the computational process for producing Automated Enhancement Technical Points will provide a blueprint for its calculation in the full Automated Technical Point Scope.

6.1.2 Types, Scopes, and Unadjusted Size Measures of Maintenance and Enhancement Work

The calculation of Automated Enhancement Points requires three well-defined scopes of measurement within the Application Scope and its component Functional and Technical Scopes. The measurement process will produce two intermediate measures of size, Automated Enhancement Function Points and Automated Enhancement Technical Points. These intermediate size measures will be combined into equivalent a measure called Automated Enhancement Points that can be used in productivity measurement and analysis programs. The scopes computational objects included in these measures are as follows:

- **Automated Enhancement Function Point Scope**—all computational objects (measured elements of data or transactional functions) within the Automated Function Points Scope that have been changed (added, modified, or deleted) between two releases of an application.

- **Automated Enhancement Technical Point Scope**—the computational objects within the Automated Technical Points Scope that have been changed (added, modified, or deleted) between two releases of an application.
- **Automated Enhancement Point Scope**—all computational objects within the combined Automated Enhancement Function Points Scope and Automated Enhancement Technical Points Scope between two releases of an application.

An automated measure will be specified for sizing the maintenance and enhancement work within each of the three scopes defined in the previous paragraph.

- **Automated Enhancement Function Points (AEFP)**—an Automated Function Point score calculated on only those computational elements within the Automated Enhancement Function Point Scope.
- **Automated Enhancement Technical Points (AETP)**—an automated measure for sizing the changes made to computational elements within the Automated Enhancement Technical Point Scope that is calculated as the sum of their Effort Complexities.
- **Automated Enhancement Points (AEP)**—an automated measure for sizing changes made to computational objects in the Automated Enhancement Point Scope and calculated from AEFP and equivalence-adjusted AETP.

The relationships between these three measures and their associated scopes of measurement are presented in Figure 6.1. The content and sizes of these three scopes are specific to a single release since the ensemble of computational objects added, modified, or deleted are different in each release. Therefore, all measures related to Automated Enhancement Points are calculated uniquely for changes in a specific release, and these measures can vary widely across releases based on differences in the scopes resulting from the changes implemented.

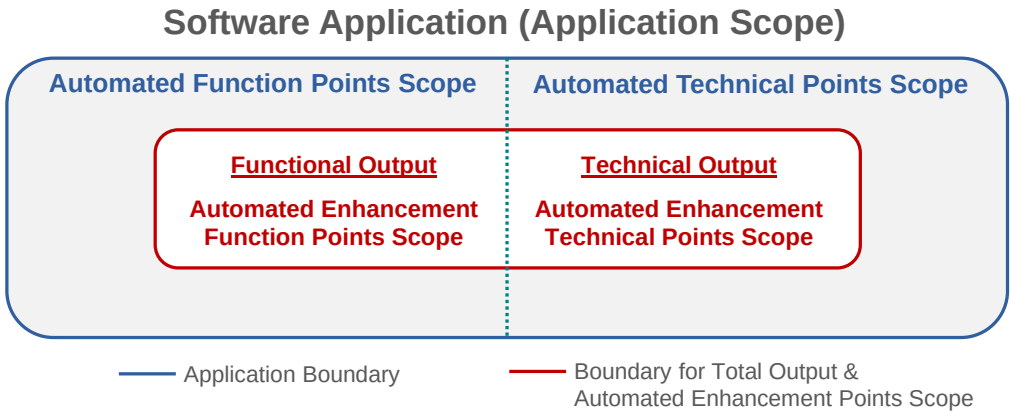

Figure 6.1. Scopes of Measurement for Components of Automated Enhancement Points

6.1.3 Adjustment Factors and Implementation Points

The problems listed in clauses 1.2.1 and 1.2.2 involve difficulties in relating maintenance and enhancement effort to software changes. These difficulties are created by differences in the complexities of the changes being implemented and the software environment in which they are implemented. These measures will be initially calculated on each Artifact involved in a change (added, modified, deleted). The complexity adjustments used in calculating the constituent measures of Automated Enhancement Points include:

- **Effort Complexity (EC)**—assesses the complexity of adding, modifying, or deleting an Artifact based on a composite score of five software metrics that assess the complexity of the software environment in which the Artifact is embedded, that is, its size, comment level, algorithmic complexity, data access complexity, and coupling.
- **Complexity Factor (CF)**—a composite score calculated from values assigned for the complexity of changes made to individual computational objects in a transaction (resp. a data entity), and then applied to adjust the Automated Enhancement Function Points of a transaction (resp. a data entity) to accurately reflect the complexity of the changes implemented, such as not double-counting the score contributions of Artifacts shared between transactions.

Transactions and data entities with the Automated Enhancement Function Point Scope are adjusted by the Complexity Factor to account for the amount by which the complexity of the changes made to them affects their implementation effort. Automated Enhancement Function Points are then calculated by multiplying the Automated Function Points of the evolved transactions and data entities by the Complexity factor.

$$\mathbf{AEFP = CF \times (\Sigma\ AFP_{changedtransactions})}$$

The problem listed in clauses 1.2.3 involve the absence of any measurement of the evolution of the technical foundation of the software. The missing measurement is addressed by:

- **Implementation Points (IP)**—a software sizing measure that accounts for the difficulty of implementing or changing computational objects by aggregating the Effort Complexity scores for Artifacts within any given scope.
- Four salient scopes are the Automated Function Point Scope, the Automated Technical Point Scope, the Automated Enhancement Function Point Scope and the Automated Enhancement Technical Point Scope of a release, leading to $\mathbf{IP_{AFP}}$, $\mathbf{IP_{ATP}}$, $\mathbf{IP_{AEFP}}$, and $\mathbf{IP_{AETP}}$ sizing measures respectively.
- $\mathbf{IP_{AFP}}$ will be used later in the process, in conjunction with **AFP** value, to compute the Equivalence Ratio between **AFP** and $\mathbf{IP_{AFP}}$ in this revision of this software, allowing to get an **AFP equivalent** value for $\mathbf{IP_{AETP}}$

6.1.4 Calculation of Automated Enhancement Points

At this stage Implementation Points associated with Automated Technical Enhancement Points and Automated Enhancement Function Points are not comparable since Automated Enhancement Function Points are calculated differently than Implementation Points of Automated Enhancement Technical Points. The next step is to adjust the Implementation Points associated with Automated Enhancement Technical Points to be statistically equivalent to Automated Enhancement Function Points using an equivalence ratio.

Equivalence Ratio (ER)—calculated by dividing the Automated Function Points score of an application by the Implementation Points score associated with all computational objects within the Automated Function Point Scope of the application.

$$\mathbf{ER = AFP\ /\ IP_{AFP}}$$

To improve their accuracy, organizations should collect Equivalence Ratio values over the various releases of the applications in their portfolio, along with other descriptive information about type of application, technology, architecture, platform, etc., in order to develop standard Equivalence Ratios for distinct classes of applications

The Equivalence Ratio is applied to the Implementation Points derived from the Automated Enhancement Technical Points score to create a unit of measure that is statistically equivalent to the Implementation Points measure derived from Automated Enhancement Function Points.

$$\mathbf{AETP = ER \times IP_{AETP}}$$

Once applied, the two equivalent scores for Implementation Points can be combined to produce the Automated Enhancement Point score:

$$\mathbf{AEP = AEFP + AETP}$$

Consequently,

- Effort expended on changes within the Automated Enhancement Function Points Scope should be correlated with the Automated Enhancement Function Points.
- Effort expended on changes within the Automated Enhancement Technical Points Scope should be correlated with the Automated Enhancement Technical Points.
- Effort expended on changes within the Automated Enhancement Points Scope should be correlated with Automated Enhancement Points which is calculated by summing the Automated Enhancement Function Points with the equivalence-adjusted Automated Enhancement Technical Points.

An Automated Enhancement Points score is expressed as a form of Automated Function Point measure since

- Automated Enhancement Function Points are Automated Function Points multiplied by a simple modifier
- Automated Enhancement Technical Points are Implementation Points multiplied by an Implementation Points-to-Automated Function Points equivalence ratio.

In organizations without any Function Point history / affinity, or organizations whose software is not a natural match for Automated Function Point computation (such as scientific computing), the sum of $\mathbf{IP_{AEFP}}$ and $\mathbf{IP_{AETP}}$ proves a valuable alternative. $\mathbf{IP_{AEFP} + IP_{AETP}}$ focusses on the algorithmic part of the software evolutions.

6.2 Developing the Application Model

The calculation of Automated Enhancement Points is performed between two releases or revisions of the software, which will be called V1 and V2, V2 being the more recent of the two releases. Both releases will be analyzed to create an Application Model of the software for each release. The Application Model is composed of computational objects in the source code and their relationships. Some of these computational objects contain processing rules and logic. Some, but not all of these computational objects will be called Artifacts (for instance, data elements are not Artifacts), and for the

purpose of this specification, Artifacts cannot contain other Artifacts. Appendix A lists Data Types that are considered Artifacts in many of the most frequently used IT systems programming languages.

The Application Model will be used to define the Automated Function Points Transaction Implementation Scope of each transaction, i.e., the list of computational objects referenced by the AFP transaction entry points, as identified in the AFP specifications (§ 6.5.2). The union of all Automated Function Points Transaction Implementation Scopes will compose the Automated Function Points Scope of the application.

6.2.1 Application Model Elements

The Application Model is produced by analyzing the source code of the application to be sized. It shall contain the computational elements of the application.
In addition to the computational elements required to Automated Function Points, computations in this specification also require:

- Building directed graphs representing the direct dependencies of objects starting from the transactional entry points (no longer limited to the sole directed graphs connecting data elements and transactional entry points).
- Identification of all named computational elements containing processing logic (Artifacts) within and outside of the directed graphs
- Identification of all named computational elements containing processing logic (Artifacts) that are shared by multiple directed graphs
- Identification of all computational elements with an evolvedTo/evolvedFrom relationship, i.e., code elements V1 that are found in V2 as an evolved version of the computational element or go unchanged.

6.2.2 Detection of Transactional and Data Functions

Unchanged from AFP 1.0 section 6.5

6.2.3 Detection of AFP transaction implementation scope

AFP transaction implementation scopes are directed graphs:

- starting from the software User Interface (or public API),
- using field use, state change, method and function invocation, class inheritance, and interface implementation dependency relationships (that is, dependencies as defined in AFP 1.0 section 6.5),
- and ending with data functions, software boundary, or the absence of a further dependency relationship.

Contrary to AFP 1.0 specification requirements for which the existence of at least one code path between the transaction entry points and the data functions were enough, the comprehensiveness of the content of these directed graphs is critical for computing some of the intermediate measures in this specification and will impact the final calculation of Automated Enhancement Points.

6.2.4 Detection of Artifacts

Artifacts are all computational objects with a name that contain application logic to support the software processing. These are primarily the Methods, Functions, and Procedures from AFP 1.0 section 6.5. In addition:

- An Artifact cannot belong to another Artifact so as to avoid any duplicate count of the value of their code metrics (e.g.: counting SQL complexity of both parent and child Artifacts would over-estimate the SQL complexity present in the software).
- Preprocessor directives are not considered as Artifacts even though some of them could alter the behavior of real Artifacts.

Each Artifact will be analyzed in both releases to determine whether it is:

- Added—which exists in release V2 while it didn't exist in release V1
- Deleted —existed in release V1 while it doesn't exist in release V2
- Modified—exists in both releases but whose checksum changed between release V1 and release V2, creating the evolvedFrom/evolvedTo association needed for historical measurement *(SMM clause §17.1)*

6.2.5 Code Metric Requirements to Compute Artifact Effort Complexity

In order to compute Artifact Effort Complexity (EC), the following code metrics are used to feed the EC formula:

- Fan-In
- McCabe Cyclomatic Complexity
- Number of code lines, excluding comments and empty lines
- Number of comment lines

When SQL queries are involved, the following additional code metrics are used:

- Number of SQL tables involved
- Number of subqueries
- Usage of 'group by' statement
- Usage of 'update' statement
- Number of columns returned by select statements

6.3 Measurement Calculations in the whole Application Scope

For each Artifact within the Application Scope, its Effort Complexity will be computed as follow.

The Effort Complexity EC of an Artifact is a function of the following five parameters:

1. Artifact Cyclomatic Complexity, as defined by McCabe, leads to Artifact assignment to one of the four categories below, using thresholds.
2. Artifact Size, as measured by the number of Lines Of Code, leads to Artifact assignment to one of the four categories below, using thresholds.
3. Artifact Comment level, as measured by the number of Lines Of Code and the number of Lines Of Comment Code, leads to Artifact assignment to one of the four categories below, using thresholds
4. Artifact Coupling, as measured by the Fan-In, leads to Artifact assignment to one of the four categories below, using thresholds.
5. Artifact SQL Complexity, based on the number of SQL tables involved in the SQL queries, the number of subqueries, the presence of a group by statement, the update nature, and the

number of columns returned by select statements, leads to Artifact assignment to one of the four categories below, using thresholds.

6.3.1 Artifact Cyclomatic Complexity category assignment

Artifact will be assigned one of the following four categories, based on the comparison of their Cyclomatic Complexity value and thresholds:

- Low Complexity category:
  Cyclomatic Complexity < 5
- Moderate Complexity category:
  Cyclomatic Complexity >= 5 && Cyclomatic Complexity < 15
- High Complexity category:
  Cyclomatic Complexity >= 15 && Cyclomatic Complexity < 30
- Very High Complexity category:
  Cyclomatic Complexity >= 30

6.3.2 Artifact Size category assignment

Artifact will be assigned one of the following four categories, based on the comparison of their number of Lines of Code and thresholds:

- Small Size category:
  Lines Of Code < 10
- Average Size category:
  Lines Of Code >= 10 && Lines Of Code < 50
- Large Size category:
  Lines Of Code >= 50 && Lines Of Code < 200
- Very Large Size category:
  Lines Of Code >= 200

6.3.3 Artifact Comment Level category assignment

Artifact will be assigned one of the following four categories, based on the comparison of their Comment level and thresholds:

- High Comment/Code category:
  Comment Level > 15%
- Average Comment/Code category:
  Comment Level <= 15% && Comment Level > 7%
- Low Comment/Code category:
  Comment Level <= 7% && Comment Level > 3%
- Very Low Comment/Code category:
  Comment Level <= 3%

6.3.4 Artifact Coupling category assignment

Artifact will be assigned one of the following four categories, based on the comparison of their Artifact Coupling value and thresholds:

- Low Coupling category:
  Number of Callers < 4

- Moderate Coupling category:
  Number of Callers >= 4 && Number of Callers < 10
- High Coupling category:
  Number of Callers >= 10 && Number of Callers < 30
- Very High Coupling category:
  Number of Callers >= 30

6.3.5 Artifact SQL Complexity category assignment

Artifact will be assigned one of the following four categories, based on the comparison of their Artifact SQL Complexity value and thresholds:

- Low SQL Complexity category:
  SQL Artifact Complexity < 10
- Moderate SQL Complexity category:
  SQL Artifact Complexity >= 10 && SQL Artifact Complexity < 40
- High SQL Complexity category:
  SQL Artifact Complexity >= 40 && SQL Artifact Complexity < 70
- Very High SQL Complexity category:
  SQL Artifact Complexity >= 70

With, the Artifact SQL Complexity rated from 0 to 100, 0 for lowest Complexity and 100 for Highest complexity.

SQL Artifact Complexity =
50 if Artifact with a Query on more than 4 tables
+ 10 if Artifact with a Subquery
+ 10 if Artifact with a GROUP BY
+ 10 if Artifact with a Complex SELECT clause
+ 10 if Artifact with an UPDATE statement
+ 10 if Artifact with Raw SQL Complexity Higher than 30

6.3.6 Artifact Effort Complexity category assignment

Once the above assignments to categories are done for an artifact, the overall Effort Complexity score is then calculated as follows:

- An Artifact falls into the Very High Effort Complexity category, when ONE of the following is true:
  - Cyclomatic Complexity is Very High
  - SQL Complexity is Very High
  - Artifact Granularity is Very High AND Lack of Comment index is Very High AND Artifact Coupling is Very High
- An Artifact falls into the High Effort Complexity category, when ONE of the following is true:
  - Cyclomatic Complexity is High
  - SQL Complexity is High
  - Cyclomatic Complexity is Moderate AND SQL Complexity is Moderate AND (Artifact Granularity is Very High OR Lack of Comment is Very High OR Artifact Coupling is Very High)
  - Artifact Granularity is High AND Lack of Comment is High AND Artifact Coupling is High

- An Artifact falls into the Moderate Effort Complexity category, when ONE of the following is true::
    - Cyclomatic Complexity is Moderate
    - SQL Complexity is Moderate
    - Cyclomatic Complexity is Low AND SQL Complexity is Low AND (Artifact Granularity is High OR Lack of Comment is High OR Artifact Coupling is High)
    - Artifact Granularity is Moderate AND Lack of Comment is Moderate AND Artifact Coupling is Moderate
- An Artifact falls into the Low Effort Complexity category, otherwise.

6.3.7 Artifact Effort Complexity final value

Each Artifact Effort Complexity category gets an Effort Complexity value, which can be overridden for specific technologies. Default are as follow:

- Very High Effort Complexity category: 1.2
- High Effort Complexity category: 0.7
- Moderate Effort Complexity category: 0.2
- Low Effort Complexity category: 0.1

6.4 Measurement Calculations in the Automated Function Points Transaction and Data Implementation Scope

Both releases will be analyzed and measured according to the Automated Function Point specification to identify transaction functions and data functions *(AFP clauses 6.5.2 and 6.5.3)*. The analysis of transactional and data functions will involve the following steps

- An **added transaction or data entity** exists in release V2 but didn't exist in V1, and its value in Automated Function Points will be denoted as $\mathbf{AFP_{V2}}$, in essence its AFP value calculated in V2.
- A **deleted transaction or data entity** no longer exists in release V2 but existed in release V1, and its size in Automated Function Points will be the denoted $\mathbf{AFP_{V1}}$, in essence its AFP value calculated in V1.
- A **split data entity** is a data entity whose DET, once part of the same data entity in release V1, are part of more than one data entity in release V2
- A **merged data entity** is a data entity whose DET were part of more than one data entity in release V1
- A **data entity with changed type** is a data entity which was identified as an Internal Logical File in release V1 (resp. an External Input File) and is identified as an External Input File in release v2 (resp. an Internal Logical File)
- A **modified transaction** exists when one or more of the computational objects in its Implementation Scope is modified, added to the processing flow, or removed from the processing flow. Computational object modification status is based on their source code checksum, as measured in releases V1 and V2. The AFP value of modified transactions will be denoted as $\mathbf{AFP_{V2}}$ in essence its AFP value calculated in V2.

- A **modified data entity** exists when one or more of the computational objects supporting the DET is modified, and when the data entity is not a split data entity or a merged data entity. The AFP value of modified data entities will be denoted as $\mathbf{AFP_{V2}}$ in essence its AFP value calculated in V2.
- The amount of modification performed on an evolved transaction or data entity will be accounted by the Complexity Factor (CF), which will be used to adjust the transaction's or data entity's AFP value.
- This Complexity Factor will also be used to adjust AFP values for added and deleted transactions and data entities.
- The Automated Enhancement Function Point score for transactions and data entities within the Automated Enhancement Function Point Scope will be the sum of their AFP values adjusted by the Complexity Factor.
- The Automated Enhancement Function Point score for the software revision, that is, for all evolved transactions and data entities is calculated as: $\mathbf{AEFP = \Sigma\ (CF_{added/modified} * AFP_{V2}) + \Sigma\ (CF_{deleted} * AFP_{V1})}$

The Complexity Factor (CF) for transactional functions is determined as follows:

- Added transaction—CF value is always 1
- Deleted transaction—CF value is always 0.4
- Modified transaction—CF values for transactions that did not contain shared Artifacts that were changed are presented in Table 2 and are based on two input parameters:
  a. **Evolved Effort Complexity**—the share of the Effort Complexity of the full transaction that was affected by the Effort Complexities of the changed Artifacts.
  b. **Effort Complexity Variation**—a percentage of the total Effort Complexity of the full transaction, to account for the Effort Complexity that was added to or removed from changed Artifacts.

Evolved Effort Complexity ($EC_{evolved}$)	Effort Complexity Variation ($EC_{variation}$)			
	⅓ (x 100%)	⅔ (x 100%)	≤ 100%	> 100%
⅓ x 100%	0.25	0.50	0.75	1.00
⅔ x 100%	0.50	0.75	1.00	1.25
≤ 100%	0.75	1.00	1.25	1.50
> 100%	1.00	1.25	1.50	1.75

Table 2. Determination of Effort Complexity Variation

- CF values for transactions that did contain shared Artifacts that were changed are determined as follows:
  a. If only a few of the modified Artifacts in a transaction are shared components, CF = 0.25
  b. If more than 75% of the modified Artifacts are shared components, CF is capped at 0.50
  c. If more than 50% of the modified Artifacts are shared components, CF is capped at 0.75

The Complexity Factor (CF) for data functions is determined as follows:

- Added data—CF value is always 1 for AFP calculated on added data, excluding 'added data' that result from a split and merge activity between the two releases
- Deleted data—CF value is always 0.4 for AFP calculated on deleted data, excluding 'delete data' resulting from a split and merge activity between the two releases
- Merged data—CF value is always 0.4 for AFP calculated on merged data
- Split data—CF value is always 0.4 for AFP calculated on split data
- Data with changed type—CF value is always 0.4 for AFP calculated on data with changed type (e.g., EIF ⇒ ILF) but no change of DET score
- Modified data—CF value of a modified data function is based on one input parameter as presented in Table 3:

% of DETs affected by the change	⅓ (x 100%)	⅔ (x 100%)	≤ 100%	> 100%
Modified Data CF	0.25	0.50	0.75	1.00

Table 3. Complexity Factor for Modified Data

6.5 Measurement Calculations in the Automated Enhancement Technical Points Scope

The Implementation Points associated with Automated Enhancement Technical Points are calculated as the sum of the Effort Complexities of all the Artifacts within the Automated Enhancement Technical Points Scope, that is:

$$IP_{AETP} = \Sigma\ (EC_{AETP})$$

The Implementation Points associated with Automated Enhancement Technical Points are multiplied by the Equivalence Ratio to transform them into a measure that is equivalent to the Automated Enhancement Function Points.

6.6 Outline of the Automated Enhancement Points Calculation Process

For both V1 and V2 releases:

- Collect required input
- Generate the application model
- Compute AFP scopes and metrics
  - AFP detection in application model (according to formal/2013-01-02)
  - AFP complexity and sizing (according to OMG AFP)
  - AFP Implementation Scopes in application model (new: not required in OMG AFP) (leading implicitly to computation of the ATP scope, complementary to the AFP scope in the whole software)
  - Shared AFP Elements scope

- Compute Implementation scopes and metrics
    - Artifacts EC in application model
    - IP_{AFP}
    - IP_{ATP}

For V2 release, using V1 release as its previous release:

- Generate Artifacts evolution scopes
    - Added, Updated, and Deleted Artifacts between application models
    - Added, Updated, and Deleted AFP Artifacts
    - Added, Updated, and Deleted Shared ATP Artifacts
- Compute evolved implementation scopes and metrics
    - $Scope_{AEFP}$
    - $Scope_{AETP}$
- Identify evolved AFP
    - Added, Updated, and Deleted Transactional AFP
    - Added, Updated, Deleted, Merged, Split, 'with changed type' Data AFP
- Compute complexity factor for evolved AFP
    - $EC_{variation}$, $EC_{evolved}$, EC_{shared}
- Compute resulting AEFP
- Compute AETP
    - EC for all Artifacts in $Scope_{AETP}$
- Compute functional equivalent form of IP_{AEFP}
    - ER from AFP and IP_{AFP}
    - $AETP_{EQ}$ from ER and IP_{AETP}
- Compute AEP from AEFP and $AETP_{EQ}$

7. Determine Software Enhancement Productivity (Normative)

The normative clause 7 is supported by the content of the machine consumable XMI file: http://www.omg.org/spec/AEP/20151109/AutomatedEnhancementPoints.xmi

7.1 Representation of the Application Model

7.1.1 Representation in KDM

In addition to the representation of Application Model used for the functional size measurement according to the AFP 1.0 specification document (chapter 7 section 7.1.1), the current specification document also requires to define the KDM elements whose instances are

- Artifacts: named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored', with non-empty 'name' attribute)
- Artifact complexity information
    - Artifact Cyclomatic Complexity (cf. SMM 1.0 chapter 19 section 19.3.2)
    - Artifact number of Lines of Code (cf. SMM 1.0 chapter 19 section 19.2.4 and specifically figure 19.13 for control element lines of code)
    - Artifact commentedness ratio (cf. SMM 1.0 chapter 19 section 19.6)
    - Artifact number of used SQL Tables: count of relational tables (data:RelationalTables) with read or write action from the Artifact (data:ReadsColumnSet or data:WritesColumnSet)
    - Artifact number of used SQL Table Columns: count of item unit data elements (code:ItemUnit) from relational tables (data:RelationalTables) involved in data read action from the Artifact (data:ReadsColumnSet)
    - Artifact number of subqueries: count of action elements performing a data write or read action (data:ReadsColumnSet or data:WritesColumnSet) nested in an action element performing a data read or write action (data:ReadsColumnSet or data:WritesColumnSet).
    - Artifact number of used "Group By" SQL statement
    - Artifact number of used "Update" SQL statement: count of data write action from the Artifact (data:WritesColumnSet)
    - Artifact Fan-In: count of inward callable relations (action:CallableRelations) where the Artifact is the target control element (to:ControlElement)
- AFP transaction implementation scope: logical block units (action:BlockUnit) composed of computational objects (code:ComputationalObject) connected via callable relations (action:CallableRelations), data relations (action:DataRelations), class relations (code:Extends), and interface relations (code:Implements) starting with a user interface (UI:UIDisplay)
- "evolvedTo/evolvedFrom" relationships as introduded in SMM 1.0 §17.1 as tracking capability over the revision or release is key to AEFP.

7.1.2 Representation in SMM

The application model representation in KDM will enable the following modeling:

- One Observation of both revisions so that the base model contains all required items
- Two Observation Scopes for this Observation to easily distinguish between the two revisions
- A set of Measures, Scopes, recognizer Operations which target a specific revision.

- A set of Measures, Scopes, recognizer Operations which target both revisions at the same time

7.1.3 Organization of the following sections

The various instances of the SMM Classes are listed:

- by sequence of computation in sections clause 0, to get a logical overview of the whole process
- by implemented Class in sections clauses 0 to 0, with their full definition details, with required parameters listed in clause 0 and references to external specifications listed in clause 0, to get a human-readable version of the machine consumable file attached to the current specification document.

7.2 SMM elements referenced in computation dependency sequence

7.2.1 Artifact level

For each Artifact, the Effort Complexity value which underlies the Implementation Point counting is computed using basis application model information, that are subsequently post-processed into a single value.

(using scope SQLQuery: Scope, that is SQL queries which reads and writes data)

- QueryNumberOfSQLTables: DirectMeasure
  To count the number of SQL tables used by a SQL query (using scope kdm:data::DataActions)

(using scope Artifact: Scope, that is, named callable computational objects such as functions, procedures, and methods)

- ArtifactMaxNumberOfSQLTablesPerQuery: CollectiveMeasure
  To compute at the Artifact level, the maximum number of SQL tables used by any of its contained SQL queries (QueryNumberOfSQLTables: DirectMeasure)
- ArtifactLinesOfCode: NamedMeasure
  To count the number of Lines of Code
- ArtifactLinesOfCommentCode: NamedMeasure
  To count the number of Lines of Comment Code
- ArtifactLinesOfCodeAndCommentCode: CollectiveMeasure
  To count the sum of the number of Lines of Code (ArtifactLinesOfCode: NamedMeasure) and the number of Lines of Comment Code (ArtifactLinesOfCommentCode: NamedMeasure)
- ArtifactCommentRatio: RatioMeasure
  To compute the ratio of Lines of Comment Code (ArtifactLinesOfCommentCode: NamedMeasure) per number of Lines of Code and Comment Code (ArtifactLinesOfCodeAndCommentCode: CollectiveMeasure)
- ArtifactCyclomaticComplexity: NamedMeasure
  To compute the Artifact's McCabe Cyclomatic Complexity
- ArtifactFanIn: DirectMeasure
  To count the number of references
- ArtifactNumberOfGroupBySQLStatement: DirectMeasure
  To count the number of "Group By" SQL statement in Artifact's SQL queries

- ArtifactNumberOfSQLSubqueries: DirectMeasure
  To count the number of sub-queries in Artifact's SQL queries
- ArtifactNumberOfUpdateSQLStatement: DirectMeasure
  To count the number of "Update" SQL statement in Artifact's SQL queries
- ArtifactNumberOfUsedSQLTableColumns: DirectMeasure
  To count the number of SQL table columns used in Artifact's SQL queries
- ArtifactNumberOfUsedSQLTables: DirectMeasure
  To count the number of SQL tables used in Artifact's SQL queries
- wArtifactMaxNumberOfSQLTablesPerQuery: RescaledMeasure
  *To compute the contribution to overall SQL complexity based on the maximal number of used SQL tables per SQL query in the Artifact (*ArtifactMaxNumberOfSQLTablesPerQuery: CollectiveMeasure *)*
- wArtifactNumberOfUsedSQLTables: RescaledMeasure
  To compute the contribution to overall SQL complexity based on the number of used SQL tables in Artifact's SQL queries (ArtifactNumberOfUsedSQLTables: DirectMeasure)
- wArtifactNumberOfUsedSQLTableColumns: RescaledMeasure
  To compute the contribution to overall SQL complexity based on the number of used SQL table columns in Artifact's SQL queries (ArtifactNumberOfUsedSQLTableColumns: DirectMeasure)
- wArtifactNumberOfSQLSubqueries: RescaledMeasure
  To compute the contribution to overall SQL complexity based on the number of sub-queries in Artifact's SQL queries (ArtifactNumberOfSQLSubqueries: DirectMeasure)
- wArtifactNumberOfGroupBySQLStatement: RescaledMeasure
  To compute the contribution to overall SQL complexity based on the number of "Group By" Statement in Artifact's SQL queries (ArtifactNumberOfGroupBySQLStatement: DirectMeasure)
- wArtifactNumberOfUpdateSQLStatement: RescaledMeasure
  To compute the contribution to overall SQL complexity based on the number of "Update" Statement in Artifact's SQL queries (ArtifactNumberOfUpdateSQLStatement: DirectMeasure)
- ArtifactSQLComplexity: CollectiveMeasure
  To compute the overall SQL complexity value by summing the various contributions (wArtifactNumberOfUpdateSQLStatement: RescaledMeasure, wArtifactNumberOfGroupBySQLStatement: RescaledMeasure, wArtifactNumberOfSQLSubqueries: RescaledMeasure, wArtifactNumberOfUsedSQLTableColumns: RescaledMeasure, wArtifactNumberOfUsedSQLTables: RescaledMeasure, wArtifactMaxNumberOfSQLTablesPerQuery: RescaledMeasure)
- SQLComplexityLevel: Ranking
  To assign a SQL complexity level (low, moderate, high, very high) based on its value (ArtifactSQLComplexity: CollectiveMeasure) and a set of thresholds (managed as parameters)
- wArtifactSQLComplexityLevel: RescaledMeasure
  To compute the SQL complexity level bit into the overall effort complexity, based on its level (SQLComplexityLevel: Ranking)
- LinesOfCodeLevel: Ranking
  To assign a Sizing level (small, medium, large, very large) based on its value (ArtifactLinesOfCode: NamedMeasure) and a set of thresholds (managed as parameters)

- wArtifactLinesOfCodeLevel: RescaledMeasure <- LinesOfCodeLevel: Ranking
  To compute the Size level bit into the overall effort complexity, based on its level (LinesOfCodeLevel: Ranking)
- LackOfCommentLevel: Ranking
  To assign a Lack of Comment level (low, moderate, high, very high) based on its value (ArtifactCommentRatio: RatioMeasure) and a set of thresholds (managed as parameters)
- wArtifactLackOfCommentLevel: RescaledMeasure <- LackOfCommentLevel: Ranking
  To compute the Lack of Comment level bit into the overall effort complexity, based on its level (LackOfCommentLevel: Ranking)
- CouplingLevel: Ranking
  To assign a SQL complexity level (low, moderate, high, very high) based on its value (ArtifactFanIn: DirectMeasure) and a set of thresholds (managed as parameters)
- wArtifactCouplingLevel: RescaledMeasure <- CouplingLevel: Ranking
  To compute the Coupling level bit into the overall effort complexity, based on its level (CouplingLevel: Ranking)
- CyclomaticComplexityLevel: Ranking
  To assign a SQL complexity level (low, moderate, high, very high) based on its value (ArtifactCyclomaticComplexity: NamedMeasure) and a set of thresholds (managed as parameters)
- wArtifactCyclomaticComplexityLevel: RescaledMeasure <- CyclomaticComplexityLevel: Ranking
  To compute the Cyclomatic Complexity level bit into the overall effort complexity, based on its level (CyclomaticComplexityLevel: Ranking)
- ArtifactEffortComplexityValue: CollectiveMeasure
  To compute an Effort Complexity binary value summing the contributing bits (wArtifactSQLComplexityLevel: RescaledMeasure, wArtifactLinesOfCodeLevel: RescaledMeasure, wArtifactLackOfCommentLevel: RescaledMeasure, wArtifactCouplingLevel: RescaledMeasure, wArtifactCyclomaticComplexityLevel: RescaledMeasure)
- ArtifactEffortComplexityIndex: RescaledMeasure <- ArtifactEffortComplexityValue: CollectiveMeasure
  To compute an aggregated index value based on the presence of specific contributing bits in the Effort Complexity value (ArtifactEffortComplexityValue: CollectiveMeasure)
- ArtifactEffortComplexityLevel: Ranking
  To turn into the Effort Complexity level (low, moderate, high, very high) the Effort Complexity index (ArtifactEffortComplexityIndex: RescaledMeasure)
- ArtifactEffortComplexity: RescaledMeasure
  To compute the final Effort Complexity, based on the level (ArtifactEffortComplexityLevel: Ranking)

7.2.2 AFP level

After processing Artifact, the application model is used to count Automated Function Points according to AFP 1.0 and then derived Automated Enhancement Function Points using additional measures of application model information

(using scope kdm:Core::Element, as defined in AFP 1.0)

- weightExternalOutput: NamedMeasure
  To compute the weight of External Ouput Transactional AFP
- weightExternalInput: NamedMeasure
  To compute the weight of External Input Transactional AFP
- weightExternalInterfaceFile: NamedMeasure
  To compute the weight of External Interface File Data AFP
- weightInternalLogicalFile: NamedMeasure
  To compute the weight of Internal Logical File Data AFP
- weightAutomatedFunctionPoints: CollectiveMeasure
  To compute an AFP size regardless of the AFP nature, by summing the results of the various AFP weights, knowing that only one result will not be null by definition (weightExternalOutput: NamedMeasure, weightExternalInput: NamedMeasure, weightExternalInterfaceFile: NamedMeasure, weightInternalLogicalFile: NamedMeasure)

(using scope AddedDataAFP: Scope, that is, EIF and ILF added in the latest revision)

- AddedDataAFPComplexityFactor: RescaledMeasure
  The adjustment Complexity Factor to use for added data AFP
- weightAddedDataFunctionPoints: BinaryMeasure
  To compute the final contribution of added data AFP into Automated Enhancement Function Point value, as the product of the associated complexity factor (AddedDataAFPComplexityFactor: RescaledMeasure) by the AFP value (weightAutomatedFunctionPoints: CollectiveMeasure)

(using scope AddedTransactionalAFP: Scope, that is, EO and EI added in the latest revision)

- AddedTransactionalAFPComplexityFactor: RescaledMeasure
  The adjustment Complexity Factor to use for added transactional AFP
- weightAddedTransactionalFunctionPoints: BinaryMeasure
  To compute the final contribution of added transactional AFP into Automated Enhancement Function Point value, as the product of the associated complexity factor (AddedTransactionalAFPComplexityFacto: RescaledMeasure) by the AFP value (weightAutomatedFunctionPoints: CollectiveMeasure)

(using scope DeletedDataAFP: Scope, that is, EIF and ILF removed from the latest revision)

- DeletedDataAFPComplexityFactor: RescaledMeasure
  The adjustment Complexity Factor to use for deleted data AFP
- weightDeletedDataFunctionPoints: BinaryMeasure
  To compute the final contribution of deleted data AFP into Automated Enhancement Function Point value, as the product of the associated complexity factor (DeletedDataAFPComplexityFactor: RescaledMeasure) by the AFP value (weightAutomatedFunctionPoints: CollectiveMeasure)

(using scope DeletedTransactionalAFP: Scope, that is, EO and EI removed from the latest revision)

- DeletedTransactionalAFPComplexityFactor: RescaledMeasure
  The adjustment Complexity Factor to use for deleted transactional AFP
- weightDeletedTransactionalFunctionPoints: BinaryMeasure
  To compute the final contribution of deleted transactional AFP into Automated Enhancement Function Point value, as the product of the associated complexity factor (

DeletedTransactionalAFPComplexityFactor: RescaledMeasure) by the AFP value (weightAutomatedFunctionPoints: CollectiveMeasure)

(using scope MergedDataAFP: Scope, that is, EIFs and ILFs from previous revision merged into single EIF and ILF in the latest revision)

- MergedDataAFPComplexityFactor: RescaledMeasure
  The adjustment Complexity Factor to use for merged data AFP
- weightMergedDataFunctionPoints: BinaryMeasure
  To compute the final contribution of merged data AFP into Automated Enhancement Function Point value, as the product of the associated complexity factor (MergedDataAFPComplexityFactor: RescaledMeasure) by the AFP value (weightAutomatedFunctionPoints: CollectiveMeasure)

(using scope SplitDataAFP: Scope, that is, single EIF and ILF from previous revision split into EIFs and ILFs in the latest revision)

- SplitDataAFPComplexityFactor: RescaledMeasure
  The adjustment Complexity Factor to use for split data AFP
- weightSplitDataFunctionPoints: BinaryMeasure
  To compute the final contribution of added data AFP into Automated Enhancement Function Point value, as the product of the associated complexity factor (SplitDataAFPComplexityFactor: RescaledMeasure) by the AFP value (weightAutomatedFunctionPoints: CollectiveMeasure)

(using scope UpdatedTransactionalAFP_Latest: Scope, that is, EO and EI with evolved implementation artifacts as defined by AFPTransactionImplementationArtifacts: OCLOperation)

- EffortComplexityTotalInLatest: CollectiveMeasure
  To compute the total Effort Complexity of updated transaction AFP in latest revision, summing Artifacts's Effort Complexity (ArtifactEffortComplexity: RescaledMeasure) on Artifacts from its implementation scope in latest revision

(using scope UpdatedTransactionalAFP_Previous: Scope, that is, implementation artifacts as defined by AFPTransactionImplementationArtifacts: OCLOperation in the previous revision)

- EffortComplexityTotalInPrevious: CollectiveMeasure
  To compute the total Effort Complexity of updated transaction AFP in previous revision, summing Artifacts's Effort Complexity (ArtifactEffortComplexity: RescaledMeasure) on Artifacts from its implementation scope in previous revision

(using scope UpdatedTransactionalAFP: Scope)

- EffortComplexityTotalVariation: BinaryMeasure
  To compute the total Effort Complexity net variation, as the difference between Its latest value (EffortComplexityTotalInLatest: CollectiveMeasure) and its previous value (EffortComplexityTotalInPrevious: CollectiveMeasure)
- RatioEffortComplexityTotalVariation: RatioMeasure
  To compute the ratio of the Effort Complexity net variation (EffortComplexityTotalVariation: BinaryMeasure) divided by the total Effort Complexity value in previous revision (EffortComplexityTotalInPrevious: CollectiveMeasure)

- wRatioEffortComplexityTotalVariation: RescaledMeasure
  To compute the contribution of the ratio of Effort Complexity variation (RatioEffortComplexityTotalVariation: RatioMeasure) into the overall Complexity Factor

(using scope ArtifactInUpdatedTransactionalAFP_Added: Scope, that is, added implementation artifacts as defined by AFPTransactionImplementationArtifacts: OCLOperation from updated transactional AFP)

- EffortComplexityAdded: CollectiveMeasure
  To compute the Effort Complexity of added Artifacts in updated transactional AFP, summing their Effort Complexity (ArtifactEffortComplexity: RescaledMeasure)

(using scope ArtifactInUpdatedTransactionalAFP_Deleted: Scope, that is, deleted implementation artifacts as defined by AFPTransactionImplementationArtifacts: OCLOperation from updated transactional AFP)

- EffortComplexityDeleted: CollectiveMeasure
  To compute the Effort Complexity of deleted Artifacts in updated transactional AFP, summing their Effort Complexity (ArtifactEffortComplexity: RescaledMeasure)

(using scope ArtifactInUpdatedTransactionalAFP_Updated: Scope, that is, updated implementation artifacts as defined by AFPTransactionImplementationArtifacts: OCLOperation from updated transactional AFP)

- EffortComplexityUpdated: CollectiveMeasure
  To compute the Effort Complexity of updated Artifacts in updated transactional AFP, summing their Effort Complexity (ArtifactEffortComplexity: RescaledMeasure)

(using scope UpdatedTransactionalAFP: Scope)

  - EffortComplexityProcessed: CollectiveMeasure
    To compute the total Effort Complexity of evolved Artifacts in updated transactional AFP, summing the added, deleted, and updated values (EffortComplexityAdded: CollectiveMeasure, EffortComplexityUpdated: CollectiveMeasure, EffortComplexityDeleted: CollectiveMeasure)
- RatioEffortComplexityProcessed: RatioMeasure
  To compute the ratio of the Effort Complexity proccessed (EffortComplexityProcessed: CollectiveMeasure) divided by the total Effort Complexity value in previous revision (EffortComplexityTotalInPrevious: CollectiveMeasure)
- wRatioEffortComplexityProcessed: RescaledMeasure
  To compute the contribution of the ratio of Effort Complexity processed (RatioEffortComplexityProcessed: RatioMeasure) into the overall Complexity Factor
- sRatioEffortComplexity: CollectiveMeasure
  To compute the sum of the two Effort Complexity contributions (wRatioEffortComplexityTotalVariation: RescaledMeasure, wRatioEffortComplexityProcessed: RescaledMeasure)
- RawUpdatedTransactionalAFPComplexityFactor: RescaledMeasure
  To compute a raw Complexity Factor for updated transactional AFP, based on the sum of Effort Complexity contributions (sRatioEffortComplexity: CollectiveMeasure)

(using scope SharedArtifactInUpdatedTransactionalAFP_Added: Scope, that is, shared added implementation artifacts as defined by AFPTransactionImplementationArtifacts: OCLOperation from updated transactional AFP)

- SharedEffortComplexityAdded: CollectiveMeasure
  To compute the Effort Complexity of shared added Artifacts in updated transactional AFP, summing their Effort Complexity (ArtifactEffortComplexity: RescaledMeasure)

(using scope SharedArtifactInUpdatedTransactionalAFP_Deleted: Scope, that is, shared deleted implementation artifacts as defined by AFPTransactionImplementationArtifacts: OCLOperation from updated transactional AFP)

- SharedEffortComplexityDeleted: CollectiveMeasure
  To compute the Effort Complexity of shared deleted Artifacts in updated transactional AFP, summing their Effort Complexity (ArtifactEffortComplexity: RescaledMeasure)

(using scope SharedArtifactInUpdatedTransactionalAFP_Updated: Scope, that is, shared updated implementation artifacts as defined by AFPTransactionImplementationArtifacts: OCLOperation from updated transactional AFP)

- SharedEffortComplexityUpdated: CollectiveMeasure
  To compute the Effort Complexity of shared updated Artifacts in updated transactional AFP, summing their Effort Complexity (ArtifactEffortComplexity: RescaledMeasure)

(using scope UpdatedTransactionalAFP: Scope)

- SharedEffortComplexityProcessed: CollectiveMeasure
  To compute the total Effort Complexity of evolved Artifacts in updated transactional AFP, summing the added, deleted, and updated values (SharedEffortComplexityAdded: CollectiveMeasure, SharedEffortComplexityUpdated: CollectiveMeasure, SharedEffortComplexityDeleted: CollectiveMeasure)
  RatioSharedEffortComplexity: RatioMeasure
  To compute the ratio of the shared Effort Complexity proccessed (SharedEffortComplexityProcessed: CollectiveMeasure) divided by the Effort Complexity processed (EffortComplexityProcessed: CollectiveMeasure)
- capRatioSharedEffortComplexity: RescaledMeasure
  To compute the capping value for the Complexity Factor of updated transactional AFP due to the sharing of Artifacts, based on the ratio of shared Effort Complexity (RatioSharedEffortComplexity: RatioMeasure)
- UpdatedTransactionalAFPComplexityFactor: CollectiveMeasure
  To compute the final value for the Complexity Factor of updated transactional AFP, as the minimum of the raw value (RawUpdatedTransactionalAFPComplexityFactor: RescaledMeasure) and the capping value (capRatioSharedEffortComplexity: RescaledMeasure)
- weightUpdatedTransactionalFunctionPoints: BinaryMeasure
  To compute the final contribution of updated transactional AFP into Automated Enhancement Function Point value, as the product of the associated complexity factor (UpdatedTransactionalAFPComplexityFactor: RescaledMeasure) by the AFP value (weightAutomatedFunctionPoints: CollectiveMeasure)

(using scope DETInUpdatedDataAFP_Added: Scope, that is, added DET owned by updated data AFP)

- AddedDETInLatest: Counting
  To count added DET in updated data AFP

(using scope DETInUpdatedDataAFP_Updated: Scope, that is, updated DET owned by updated data AFP)

- UpdatedDETInLatest: Counting
  To count updated DET in updated data AFP

(using scope DETInUpdatedDataAFP_Deleted: Scope, that is, deleted DET owned by updated data AFP)

- DeletedDETInLatest: Counting
  To count deleted DET in updated data AFP

(using scope UpdatedDataAFP: Scope, that is, EIF and ILF with evolved implementation as defined by AFPDataImplementationScope: OCLOperation)

- EvolvedDETInLatest: CollectiveMeasure
  To count evolved DET in updated data AFP as the sum of added, updated, and deleted DET (AddedDETInLatest: Counting, UpdatedDETInLatest: Counting, DeletedDETInLatest: Counting)

(using scope DETInUpdatedDataAFP_Previous: Scope, that is, DET owned by updated data AFP in previous revision)

- TotalDETInPrevious: Counting
  To count the total number of DET in updated data AFP in previous revision

(using scope UpdatedDataAFP: Scope)

- DETChangeRatio: RatioMeasure
  To compute the ratio of changed DET as the count of evolved DET (EvolvedDETInLatest: CollectiveMeasure) divided by the total number of DET in previous revision (TotalDETInPrevious: Counting)
- UpdatedDataAFPComplexityFactor: RescaledMeasure
  To compute the Complexity Factor of updated data AFP, based on the DET Change ratio (DETChangeRatio: RatioMeasure)
- weightUpdatedDataFunctionPoints: BinaryMeasure
  To compute the final contribution of updated data AFP into Automated Enhancement Function Point value, as the product of the associated complexity factor (UpdatedDataAFPComplexityFactor: RescaledMeasure) by the AFP value (weightAutomatedFunctionPoints: CollectiveMeasure)

7.2.3 Revision level

Last, at the revision(s) level, measures leveraging Artifact- and AFP-level measures are the following:
(using scope LatestRevision: Scope)

- AutomatedFunctionPoints_Latest: CollectiveMeasure
  To compute the total Automated Function Point value in the latest revision, by summing all the Automated Function Point contributions (weightAutomatedFunctionPoints: CollectiveMeasure)

(using scope LatestAFPRevision: Scope)

- ImplementationPoints_LatestAFPScope: CollectiveMeasure
  To compute the Implementation Point value of the AFP Scope in the latest revision, by summing all the Artifact Effort Complesity values (ArtifactEffortComplexity: RescaledMeasure)

(using scope LatestRevision: Scope)

- EquivalenceRatio_Latest: RatioMeasure
  To compute the Equivalence Ratio as the total Automated Function Point value (AutomatedFunctionPoints_Latest: CollectiveMeasure) divided by the the total Implementation Point value of the AFP Scope (ImplementationPoints_LatestAFPScope: CollectiveMeasure) in the latest revision

(using scope LatestATPRevision: Scope)

- ImplementationPoints_LatestATPScope: CollectiveMeasure
  To compute the Implementation Point value of the ATP Scope in the latest revision, by summing all the Artifact Effort Complesity values (ArtifactEffortComplexity: RescaledMeasure)

(using scope EvolvedAFPArtifacts: Scope)

- ImplementationPoints_EvolvedAFPArtifacts: CollectiveMeasure
  To compute the Implementation Point value of the evolved AFP Scope, by summing all the Artifact Effort Complesity values (ArtifactEffortComplexity: RescaledMeasure)

(using scope EvolvedATPArtifacts: Scope)

- ImplementationPoints_EvolvedATPArtifacts: CollectiveMeasure
  To compute the Implementation Point value of the evolved ATP Scope, by summing all the Artifact Effort Complesity values (ArtifactEffortComplexity: RescaledMeasure)
- AutomatedEnhancementTechnicalPoint: BinaryMeasure
  To compute the Automated Enhancement Technical Point value as the product of the Equivalence Ratio (EquivalenceRatio_Latest: RatioMeasure) by the Implementation Point value of evolved ATP Scope (ImplementationPoints_EvolvedATPArtifacts: CollectiveMeasure)

(using scope LatestRevision: Scope)

- AutomatedEnhancementFunctionPoint: CollectiveMeasure
  To compute the total Automated Enhancement Function Point value by summing all contributions of evolved data and transactional AFP (weightAddedDataFunctionPoints: BinaryMeasure, weightAddedTransactionalFunctionPoints: BinaryMeasure, weightDeletedDataFunctionPoints: BinaryMeasure, weightDeletedTransactionalFunctionPoints: BinaryMeasure, weightMergedDataFunctionPoints: BinaryMeasure, weightSplitDataFunctionPoints: BinaryMeasure, weightDataWithChangedTypeFunctionPoints: BinaryMeasure, weightUpdatedDataFunctionPoints: BinaryMeasure, weightUpdatedTransactionalFunctionPoints: BinaryMeasure)

7.3 Instances of BinaryMeasure Class

BinaryMeasure AutomatedEnhancementTechnicalPoint

ID :
AutomatedEnhancementTechnicalPoint
Description :

Automated Enhancement Function Points equivalent of evolved Artifacts of the ATP Scope
Functor :
times
Unit :
AutomatedFunctionPointEquivalent
Scope :
LatestRevision
BaseMeasure1Relationship :
AutomatedEnhancementTechnicalPoint_to_EquivalenceRatio_Latest
BaseMeasure2Relationship :
AutomatedEnhancementTechnicalPoint_to_ImplementationPoints_EvolvedATPArtifacts
Trait :
TechnicalEnhancementSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightDataFunctionPointsWithChangedType

ID :
weightDataFunctionPointsWithChangedType
Description :
Weight of Updated Data AFP
Functor :
times
Unit :
AutomatedEnhancementFunctionPoint
Scope :
UpdatedDataAFP
BaseMeasure1Relationship :
weightDataFunctionPointsWithChangedType_to_DataAFPWithChangedTypeComplexityFactor
BaseMeasure2Relationship :
weightDataFunctionPointsWithChangedType_to_weightAutomatedFunctionPoints
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightMergedDataFunctionPoints

ID :
weightMergedDataFunctionPoints
Description :
Weight of Merged Data AFP
Functor :
times
Unit :
AutomatedEnhancementFunctionPoint
Scope :

MergedDataAFP
BaseMeasure1Relationship :
weightMergedDataFunctionPoints_to_MergedDataAFPComplexityFactor
BaseMeasure2Relationship :
weightMergedDataFunctionPoints_to_weightAutomatedFunctionPoints
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightSplitDataFunctionPoints

ID :
weightSplitDataFunctionPoints
Description :
Weight of Split Data AFP
Functor :
times
Unit :
AutomatedEnhancementFunctionPoint
Scope :
SplitDataAFP
BaseMeasure1Relationship :
weightSplitDataFunctionPoints_to_SplitDataAFPComplexityFactor
BaseMeasure2Relationship :
weightSplitDataFunctionPoints_to_weightAutomatedFunctionPoints
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightUpdatedDataFunctionPoints

ID :
weightUpdatedDataFunctionPoints
Description :
Weight of Updated Data AFP
Functor :
times
Unit :
AutomatedEnhancementFunctionPoint
Scope :
UpdatedDataAFP
BaseMeasure1Relationship :
weightUpdatedDataFunctionPoints_to_UpdatedDataAFPComplexityFactor
BaseMeasure2Relationship :
weightUpdatedDataFunctionPoints_to_weightAutomatedFunctionPoints
Trait :

DataEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightDeletedDataFunctionPoints

ID :
weightDeletedDataFunctionPoints
Description :
Weight of Deleted Data AFP
Functor :
times
Unit :
AutomatedEnhancementFunctionPoint
Scope :
DeletedDataAFP
BaseMeasure1Relationship :
weightDeletedDataFunctionPoints_to_DeletedDataAFPComplexityFactor
BaseMeasure2Relationship :
weightDeletedDataFunctionPoints_to_weightAutomatedFunctionPoints
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightAddedDataFunctionPoints

ID :
weightAddedDataFunctionPoints
Description :
Weight of added Data AFP
Functor :
times
Unit :
AutomatedEnhancementFunctionPoint
Scope :
AddedDataAFP
BaseMeasure1Relationship :
weightAddedDataFunctionPoints_to_AddedDataAFPComplexityFactor
BaseMeasure2Relationship :
weightAddedDataFunctionPoints_to_weightAutomatedFunctionPoints
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightUpdatedTransactionalFunctionPoints

ID :
weightUpdatedTransactionalFunctionPoints
Description :
Weight of Updated Transactional AFP
Functor :
times
Unit :
AutomatedEnhancementFunctionPoint
Scope :
UpdatedTransactionalAFP
BaseMeasure1Relationship :
weightUpdatedTransactionalFunctionPoints_to_UpdatedTransactionalAFPComplexityFactor
BaseMeasure2Relationship :
weightUpdatedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightDeletedTransactionalFunctionPoints

ID :
weightDeletedTransactionalFunctionPoints
Description :
Weight of Deleted Transactional AFP
Functor :
times
Unit :
AutomatedEnhancementFunctionPoint
Scope :
DeletedTransactionalAFP
BaseMeasure1Relationship :
weightDeletedTransactionalFunctionPoints_to_DeletedTransactionalAFPComplexityFactor
BaseMeasure2Relationship :
weightDeletedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure weightAddedTransactionalFunctionPoints

ID :
weightAddedTransactionalFunctionPoints
Description :
Weight of added Transactional AFP
Functor :

times
Unit :
AutomatedEnhancementFunctionPoint
Scope :
AddedTransactionalAFP
BaseMeasure1Relationship :
weightAddedTransactionalFunctionPoints_to_AddedTransactionalAFPComplexityFactor
BaseMeasure2Relationship :
weightAddedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

BinaryMeasure EffortComplexityTotalVariation

ID :
EffortComplexityTotalVariation
Description :
Effort Complexity net variation between latest / final revision and previous / initial revision
Functor :
difference
Unit :
ImplementationPoint
Scope :
UpdatedTransactionalAFP
BaseMeasure1Relationship :
EffortComplexityTotalVariation_to_EffortComplexityTotalInLatest
BaseMeasure2Relationship :
EffortComplexityTotalVariation_to_EffortComplexityTotalInPrevious
Trait :
FeatureEnhancementSizing
Category :
FunctionalMetrics
-

BinaryMeasure AutomatedEnhancementPoint

ID :
AutomatedEnhancementPoint
Description :
Measurement of all software enhancements (functional and technical) using a Function Point unit
Functor :
sum
Unit :
AutomatedEnhancementFunctionPoint
Scope :
LatestRevision
BaseMeasure1Relationship :

AutomatedEnhancementPoint_to_AutomatedEnhancementFunctionPoint
BaseMeasure2Relationship :
AutomatedEnhancementPoint_to_AutomatedEnhancementTechnicalPoint
Trait :
SoftwareEnhancementSizing
Category :
FunctionalMetrics
-

7.4 Instances of Characteristic Class

Characteristic TechnicalEnhancementSizing

ID :
TechnicalEnhancementSizing
Description :
Parent (if any) :
-

Characteristic FeatureEnhancementSizing

ID :
FeatureEnhancementSizing
Description :
Parent (if any) :
-

Characteristic ImplementationComplexity

ID :
ImplementationComplexity
Description :
Parent (if any) :
-

Characteristic CouplingComplexity

ID :
CouplingComplexity
Description :
Parent (if any) :
ImplementationComplexity
-

Characteristic AlgorithmicComplexity

ID :
AlgorithmicComplexity
Description :
Parent (if any) :
ImplementationComplexity
-

Characteristic SizeComplexity

ID :
SizeComplexity
Description :
Parent (if any) :
ImplementationComplexity
-

Characteristic DocumentationComplexity

ID :
DocumentationComplexity
Description :
Parent (if any) :
ImplementationComplexity
-

Characteristic DataAccessComplexity

ID :
DataAccessComplexity
Description :
Parent (if any) :
ImplementationComplexity
-

Characteristic DataSizing

ID :
DataSizing
Description :
Parent (if any) :
FeatureEnhancementSizing
-

Characteristic DataEvolutionSizing

ID :
DataEvolutionSizing
Description :
Parent (if any) :
FeatureEnhancementSizing
-

Characteristic TransactionalEvolutionSizing

ID :
TransactionalEvolutionSizing
Description :
Parent (if any) :
FeatureEnhancementSizing
-

Characteristic FeatureSizing

ID :
FeatureSizing
Description :
Parent (if any) :
-

Characteristic SoftwareEnhancementSizing

ID :
SoftwareEnhancementSizing
Description :
Parent (if any) :
-

7.5 Instances of CollectiveMeasure Class

CollectiveMeasure weightAutomatedFunctionPoints

ID :
weightAutomatedFunctionPoints
Description :
Weight of the Data or Transactional Function as defined by AFP 1.0 specifications, be it EO, EI, ILF or EIF (only one of wEO, wEI, wILF, and wEIF will not be zero by specification)
Accumulator :
sum
Unit :
AutomatedFunctionPoint
Scope :
kdm:Core::Element
BaseMeasureRelationship(s) :
weightAutomatedFunctionPoints_to_weightExternalOutput
weightAutomatedFunctionPoints_to_weightExternalInput
weightAutomatedFunctionPoints_to_weightExternalInterfaceFile
weightAutomatedFunctionPoints_to_weightInternalLogicalFile
Trait :
FeatureSizing
Category :
FunctionalMetrics
-

CollectiveMeasure ArtifactLinesOfCodeAndCommentCode

ID :
ArtifactLinesOfCodeAndCommentCode
Description :
Number of Lines of Code and Lines of Comment Code in Artifact
Accumulator :
sum
Unit :

Line
Scope :
Artifact
BaseMeasureRelationship(s) :
ArtifactLinesOfCodeAndCommentCode_to_ArtifactLinesOfCode
ArtifactLinesOfCodeAndCommentCode_to_ArtifactLinesOfCommentCode
Trait :
DocumentationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ImplementationPoints_EvolvedATPArtifacts

ID :
ImplementationPoints_EvolvedATPArtifacts
Description :
Implementation Points of evolved Artifacts from ATP Scope
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
EvolvedATPArtifact
BaseMeasureRelationship(s) :
ImplementationPoints_EvolvedATPArtifacts_to_ArtifactEffortComplexity
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ImplementationPoints_EvolvedAFPArtifacts

ID :
ImplementationPoints_EvolvedAFPArtifacts
Description :
Implementation Points of evolved Artifacts from AFP Scope
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
EvolvedAFPArtifact
BaseMeasureRelationship(s) :
ImplementationPoints_EvolvedAFPArtifacts_to_ArtifactEffortComplexity
Trait :
ImplementationComplexity
Category :
FunctionalMetrics

-

CollectiveMeasure AutomatedEnhancementFunctionPoint

ID :
AutomatedEnhancementFunctionPoint
Description :
Automated Enhancement Function Point value
Accumulator :
sum
Unit :
AutomatedFunctionPoint
Scope :
kdm:kdm::Segment
BaseMeasureRelationship(s) :
AutomatedEnhancementFunctionPoint_to_weightUpdatedTransactionalFunctionPoints
AutomatedEnhancementFunctionPoint_to_weightAddedTransactionalFunctionPoints
AutomatedEnhancementFunctionPoint_to_weightDeletedTransactionalFunctionPoints
AutomatedEnhancementFunctionPoint_to_weightUpdatedDataFunctionPoints
AutomatedEnhancementFunctionPoint_to_weightAddedDataFunctionPoints
AutomatedEnhancementFunctionPoint_to_weightDeletedDataFunctionPoints
AutomatedEnhancementFunctionPoint_to_weightSplitDataFunctionPoints
AutomatedEnhancementFunctionPoint_to_weightMergedDataFunctionPoints
AutomatedEnhancementFunctionPoint_to_weightDataFunctionPointsWithChangedType
Trait :
FeatureEnhancementSizing
Category :
FunctionalMetrics
-

CollectiveMeasure EvolvedDETInLatest

ID :
EvolvedDETInLatest
Description :
Number of evolved DET in latest / final revision
Accumulator :
sum
Unit :
Cardinal
Scope :
UpdatedDataAFP
BaseMeasureRelationship(s) :
EvolvedDETInLatest_to_AddedDETInLatest
EvolvedDETInLatest_to_UpdatedDETInLatest
EvolvedDETInLatest_to_DeletedDETInLatest
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics

-

CollectiveMeasure UpdatedTransactionalAFPComplexityFactor

ID :
UpdatedTransactionalAFPComplexityFactor
Description :
Complexity Factor to use to weight Updated Transactional AFP to get the resulting Automated Enhancement Function Points, based on the Effort Complexity Net Variation Ratio and the Processed Effort Complexity Ration, including the capping based on the Ratio of Shared Effort Complexity
Accumulator :
min
Unit :
Real
Scope :
UpdatedTransactionalAFP
BaseMeasureRelationship(s) :
UpdatedTransactionalAFPComplexityFactor_to_RawUpdatedTransactionalAFPComplexityFactor
UpdatedTransactionalAFPComplexityFactor_to_capRatioSharedEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure sRatioEffortComplexity

ID :
sRatioEffortComplexity
Description :
Sum of the weights from contributing Effort Complexity factors of Updated Transactional AFP
Accumulator :
sum
Unit :
Cardinal
Scope :
UpdatedTransactionalAFP
BaseMeasureRelationship(s) :
sRatioEffortComplexity_to_wRatioEffortComplexityProcessed
sRatioEffortComplexity_to_wRatioEffortComplexityTotalVariation
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure SharedEffortComplexityProcessed

ID :
SharedEffortComplexityProcessed
Description :

Effort Complexity of all Shared evolved Artifact of an Updated Transactional AFP
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
UpdatedTransactionalAFP
BaseMeasureRelationship(s) :
SharedEffortComplexityProcessed_to_SharedEffortComplexityAdded
SharedEffortComplexityProcessed_to_SharedEffortComplexityUpdated
SharedEffortComplexityProcessed_to_SharedEffortComplexityDeleted
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure SharedEffortComplexityDeleted

ID :
SharedEffortComplexityDeleted
Description :
Effort Complexity of Shared Deleted Artifact of an Updated Transactional AFP
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
SharedArtifactInUpdatedTransactionalAFP_Deleted
BaseMeasureRelationship(s) :
SharedEffortComplexityDeleted_to_ArtifactEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure SharedEffortComplexityUpdated

ID :
SharedEffortComplexityUpdated
Description :
Effort Complexity of Shared Updated Artifact of an Updated Transactional AFP
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
SharedArtifactInUpdatedTransactionalAFP_Updated
BaseMeasureRelationship(s) :

SharedEffortComplexityUpdated_to_ArtifactEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure SharedEffortComplexityAdded

ID :
SharedEffortComplexityAdded
Description :
Effort Complexity of Shared Added Artifact of an Updated Transactional AFP
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
SharedArtifactInUpdatedTransactionalAFP_Added
BaseMeasureRelationship(s) :
SharedEffortComplexityAdded_to_ArtifactEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure EffortComplexityTotalInLatest

ID :
EffortComplexityTotalInLatest
Description :
Effort Complexity of all Artifact of an Updated Transactional AFP in latest / final revision
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
UpdatedTransactionalAFP_Latest
BaseMeasureRelationship(s) :
EffortComplexityTotalInLatest_to_ArtifactEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure EffortComplexityProcessed

ID :
EffortComplexityProcessed

Description :
Effort Complexity of all evolved Artifact of an Updated Transactional AFP
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
UpdatedTransactionalAFP
BaseMeasureRelationship(s) :
EffortComplexityProcessed_to_EffortComplexityAdded
EffortComplexityProcessed_to_EffortComplexityUpdated
EffortComplexityProcessed_to_EffortComplexityDeleted
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure EffortComplexityTotalInPrevious

ID :
EffortComplexityTotalInPrevious
Description :
Effort Complexity of all Artifact of an Updated Transactional AFP in previous / initial revision
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
UpdatedTransactionalAFP_Previous
BaseMeasureRelationship(s) :
EffortComplexityTotalInPrevious_to_ArtifactEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure EffortComplexityDeleted

ID :
EffortComplexityDeleted
Description :
Effort Complexity of Deleted Artifact of an Updated Transactional AFP
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
ArtifactInUpdatedTransactionalAFP_Deleted

BaseMeasureRelationship(s) :
EffortComplexityDeleted_to_ArtifactEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure EffortComplexityUpdated

ID :

EffortComplexityUpdated
Description :
Effort Complexity of Updated Artifact of an Updated Transactional AFP
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
ArtifactInUpdatedTransactionalAFP_Updated
BaseMeasureRelationship(s) :
EffortComplexityUpdated_to_ArtifactEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure EffortComplexityAdded

ID :
EffortComplexityAdded
Description :
Effort Complexity of Added Artifact of an Updated Transactional AFP
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
ArtifactInUpdatedTransactionalAFP_Added
BaseMeasureRelationship(s) :
EffortComplexityAdded_to_ArtifactEffortComplexity
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

CollectiveMeasure AutomatedFunctionPoints_Previous

ID :

AutomatedFunctionPoints_Previous
Description :
Automated Function Point value in previous / initial revision
Accumulator :
sum
Unit :
AutomatedFunctionPoint
Scope :
PreviousRevision
BaseMeasureRelationship(s) :
AutomatedFunctionPoints_Previous_to_weightAutomatedFunctionPoints
Trait :
FeatureSizing
Category :
FunctionalMetrics
-

CollectiveMeasure AutomatedFunctionPoints_Latest

ID :
AutomatedFunctionPoints_Latest
Description :
Automated Function Point value in latest / final revision
Accumulator :
sum
Unit :
AutomatedFunctionPoint
Scope :
LatestRevision
BaseMeasureRelationship(s) :
AutomatedFunctionPoints_Latest_to_weightAutomatedFunctionPoints
Trait :
FeatureSizing
Category :
FunctionalMetrics
-

CollectiveMeasure ImplementationPoints_Previous

ID :
ImplementationPoints_Previous
Description :
Implementation Point value in previous / initial revision
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
PreviousRevision
BaseMeasureRelationship(s) :

ImplementationPoints_Previous_to_ImplementationPoints_PreviousAFPScope
ImplementationPoints_Previous_to_ImplementationPoints_PreviousATPScope
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ImplementationPoints_Latest

ID :
ImplementationPoints_Latest
Description :
Implementation Point value in latest / final revision
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
LatestRevision
BaseMeasureRelationship(s) :
ImplementationPoints_Latest_to_ImplementationPoints_LatestAFPScope
ImplementationPoints_Latest_to_ImplementationPoints_LatestATPScope
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ImplementationPoints_PreviousATPScope

ID :
ImplementationPoints_PreviousATPScope
Description :
Implementation Point value of ATP Scope in previous / initial revision
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
PreviousATPScope
BaseMeasureRelationship(s) :
ImplementationPoints_PreviousATPScope_to_ArtifactEffortComplexity
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ImplementationPoints_LatestATPScope

ID :
ImplementationPoints_LatestATPScope
Description :
Implementation Point value of ATP Scope in latest / final revision
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
LatestATPScope
BaseMeasureRelationship(s) :
ImplementationPoints_LatestATPScope_to_ArtifactEffortComplexity
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ImplementationPoints_PreviousAFPScope

ID :
ImplementationPoints_PreviousAFPScope
Description :
Implementation Point value of AFP Scope in previous / initial revision
Accumulator :
sum
Unit :
ImplementationPoint
Scope :
PreviousAFPScope
BaseMeasureRelationship(s) :
ImplementationPoints_PreviousAFPScope_to_ArtifactEffortComplexity
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ImplementationPoints_LatestAFPScope

ID :
ImplementationPoints_LatestAFPScope
Description :
Implementation Point value of AFP Scope in latest / final revision
Accumulator :
sum
Unit :
ImplementationPoint
Scope :

LatestAFPScope
BaseMeasureRelationship(s) :
ImplementationPoints_LatestAFPScope_to_ArtifactEffortComplexity
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ArtifactEffortComplexityValue

ID :
ArtifactEffortComplexityValue
Description :
Sum of the contributing bits of Effort Complexity factors
Accumulator :
sum
Unit :
Cardinal
Scope :
Artifact
BaseMeasureRelationship(s) :
ArtifactEffortComplexityValue_to_wArtifactCyclomaticComplexityLevel
ArtifactEffortComplexityValue_to_wArtifactSQLComplexityLevel
ArtifactEffortComplexityValue_to_wArtifactCouplingLevel
ArtifactEffortComplexityValue_to_wArtifactLinesOfCodeLevel
ArtifactEffortComplexityValue_to_wArtifactLackOfCommentLevel
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ArtifactSQLComplexity

ID :
ArtifactSQLComplexity
Description :
Overall SQL Complexity value between 0 and 100 of an Artifact
Accumulator :
sum
Unit :
Cardinal
Scope :
Artifact
BaseMeasureRelationship(s) :
ArtifactSQLComplexity_to_wArtifactMaxNumberOfSQLTablesPerQuery
ArtifactSQLComplexity_to_wArtifactNumberOfUsedSQLTables
ArtifactSQLComplexity_to_wArtifactNumberOfUsedSQLTableColumns
ArtifactSQLComplexity_to_wArtifactNumberOfSQLSubqueries

ArtifactSQLComplexity_to_wArtifactNumberOfGroupBySQLStatement
ArtifactSQLComplexity_to_wArtifactNumberOfUpdateSQLStatement
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

CollectiveMeasure ArtifactMaxNumberOfSQLTablesPerQuery

ID :
ArtifactMaxNumberOfSQLTablesPerQuery
Description :
Maximum number of used SQL tables in SQL queries of an Artifact
Accumulator :
max
Unit :
Cardinal
Scope :
Artifact
BaseMeasureRelationship(s) :
ArtifactMaxNumberOfSQLTablesPerQuery_to_QueryNumberOfSQLTables
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

7.6 Instances of Counting Class

Counting TotalDETInPrevious

ID :
TotalDETInPrevious
Description :
Number of DET in Data AFP updated from previous / initial revision to latest / final revision
Unit :
Cardinal
Scope :
DETInUpdatedDataAFP_Previous
Trait :
DataSizing
Category :
FunctionalMetrics
-

Counting DeletedDETInLatest

ID :
DeletedDETInLatest

Description :
Number of deleted DET in Data AFP updated from previous / initial revision to latest / final revision
Unit :
Cardinal
Scope :
DETInUpdatedDataAFP_Deleted
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

Counting UpdatedDETInLatest

ID :
UpdatedDETInLatest
Description :
Number of updated DET in Data AFP updated from previous / initial revision to latest / final revision
Unit :
Cardinal
Scope :
DETInUpdatedDataAFP_Updated
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

Counting AddedDETInLatest

ID :
AddedDETInLatest
Description :
Number of added DET in Data AFP updated from previous / initial revision to latest / final revision
Unit :
Cardinal
Scope :
DETInUpdatedDataAFP_Added
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

7.7 Instances of DirectMeasure Class

DirectMeasure ArtifactFanIn

ID :
ArtifactFanIn

Description :
Number of Calls or References to the Artifact
Unit :
Caller
Scope :
Artifact
Operation :
ArtifactFanIn_Operation
Trait :
CouplingComplexity
Category :
FunctionalMetrics
-

DirectMeasure ArtifactNumberOfUpdateSQLStatement

ID :
ArtifactNumberOfUpdateSQLStatement
Description :
Number of SQL Update statement in the Artifact
Unit :
Occurrence
Scope :
Artifact
Operation :
ArtifactNumberOfUpdateSQLStatement_Operation
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

DirectMeasure ArtifactNumberOfGroupBySQLStatement

ID :
ArtifactNumberOfGroupBySQLStatement
Description :
Number of Group By SQL Statement in the Artifact
Unit :
Occurrence
Scope :
Artifact
Operation :
ArtifactNumberOfGroupBySQLStatement_Operation
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

DirectMeasure ArtifactNumberOfSQLSubqueries

ID :
ArtifactNumberOfSQLSubqueries
Description :
Number of SQL Subqueries in the Artifact
Unit :
Occurrence
Scope :
Artifact
Operation :
ArtifactNumberOfSQLSubqueries_Operation
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

DirectMeasure ArtifactNumberOfUsedSQLTableColumns

ID :
ArtifactNumberOfUsedSQLTableColumns
Description :
Number of used SQL Table Columns in the Artifact
Unit :
Occurrence
Scope :
Artifact
Operation :
ArtifactNumberOfUsedSQLTableColumns_Operation
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

DirectMeasure ArtifactNumberOfUsedSQLTables

ID :
ArtifactNumberOfUsedSQLTables
Description :
Number of used SQL Tables in the Artifact
Unit :
Occurrence
Scope :
Artifact
Operation :
ArtifactNumberOfUsedSQLTables_Operation
Trait :
DataAccessComplexity
Category :

FunctionalMetrics
-

DirectMeasure QueryNumberOfSQLTables

ID :
QueryNumberOfSQLTables
Description :
Number of used SQL Tables in the Data Action
Unit :
Occurrence
Scope :
SQLQuery
Operation :
QueryNumberOfSQLTables_Operation
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

7.8 Instances of MeasureCategory Class

MeasureCategory ArchitectureMetrics

ID :
ArchitectureMetrics
Description :
Architecture Metrics (e.g., average call nesting level, deepest call nesting level, etc.)
Category (if any) :
CategoryElement (if any) :
-

MeasureCategory ProgramProcessMetrics

ID :
ProgramProcessMetrics
Description :
Program Process Metrics (e.g., Halstead, McCabe, etc.)
Category (if any) :
CategoryElement (if any) :
-

MeasureCategory FunctionalMetrics

ID :
FunctionalMetrics
Description :
Functional Metrics (e.g., functions defined in system, business data as a percentage of all data, functions in current
system that map to functions in target architecture, etc.)
Category (if any) :

CategoryElement (if any) :
-

MeasureCategory EnvironmentalMetrics

ID :
EnvironmentalMetrics
Description :
Environmental Metrics (e.g., number of screens, programs, lines of code, etc.)
Category (if any) :
CategoryElement (if any) :
-

7.9 Instance of MeasureRelationship Class

MeasureRelationship weightAutomatedFunctionPoints_to_weightExternalOutput

ID :
weightAutomatedFunctionPoints_to_weightExternalOutput
Type :
BaseMeasureRelationship
From :
weightAutomatedFunctionPoints
To :
weightExternalOutput
-

MeasureRelationship weightAutomatedFunctionPoints_to_weightExternalInput

ID :
weightAutomatedFunctionPoints_to_weightExternalInput
Type :
BaseMeasureRelationship
From :
weightAutomatedFunctionPoints
To :
weightExternalInput
-

MeasureRelationship weightAutomatedFunctionPoints_to_weightExternalInterfaceFile

ID :
weightAutomatedFunctionPoints_to_weightExternalInterfaceFile
Type :
BaseMeasureRelationship
From :
weightAutomatedFunctionPoints
To :
weightExternalInterfaceFile
-

MeasureRelationship weightAutomatedFunctionPoints_to_weightInternalLogicalFile

ID :
weightAutomatedFunctionPoints_to_weightInternalLogicalFile
Type :
BaseMeasureRelationship
From :
weightAutomatedFunctionPoints
To :
weightInternalLogicalFile
-

MeasureRelationship ArtifactLinesOfCodeAndCommentCode_to_ArtifactLinesOfCode

ID :
ArtifactLinesOfCodeAndCommentCode_to_ArtifactLinesOfCode
Type :
BaseMeasureRelationship
From :
ArtifactLinesOfCodeAndCommentCode
To :
ArtifactLinesOfCode
-

MeasureRelationship ArtifactLinesOfCodeAndCommentCode_to_ArtifactLinesOfCommentCode

ID :
ArtifactLinesOfCodeAndCommentCode_to_ArtifactLinesOfCommentCode
Type :
BaseMeasureRelationship
From :
ArtifactLinesOfCodeAndCommentCode
To :
ArtifactLinesOfCommentCode
-

MeasureRelationship ArtifactCommentRatio_to_ArtifactLinesOfCommentCode

ID :
ArtifactCommentRatio_to_ArtifactLinesOfCommentCode
Type :
base1MeasureRelationship
From :
ArtifactCommentRatio
To :
ArtifactLinesOfCommentCode
-

MeasureRelationship ArtifactCommentRatio_to_ArtifactLinesOfCodeAndCommentCode

ID :
ArtifactCommentRatio_to_ArtifactLinesOfCodeAndCommentCode
Type :

Base2MeasureRelationship
From :
ArtifactCommentRatio
To :
ArtifactLinesOfCodeAndCommentCode
-

MeasureRelationship AutomatedEnhancementTechnicalPoint_to_EquivalenceRatio_Latest

ID :
AutomatedEnhancementTechnicalPoint_to_EquivalenceRatio_Latest
Type :
Base1MeasureRelationship
From :
AutomatedEnhancementTechnicalPoint
To :
EquivalenceRatio_Latest
-

MeasureRelationship AutomatedEnhancementTechnicalPoint_to_ImplementationPoints_EvolvedATPArtifacts

ID :
AutomatedEnhancementTechnicalPoint_to_ImplementationPoints_EvolvedATPArtifacts
Type :
Base2MeasureRelationship
From :
AutomatedEnhancementTechnicalPoint
To :
ImplementationPoints_EvolvedATPArtifacts
-

MeasureRelationship ImplementationPoints_EvolvedATPArtifacts_to_ArtifactEffortComplexity

ID :
ImplementationPoints_EvolvedATPArtifacts_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
ImplementationPoints_EvolvedATPArtifacts
To :
ArtifactEffortComplexity
-

MeasureRelationship ImplementationPoints_EvolvedAFPArtifacts_to_ArtifactEffortComplexity

ID :
ImplementationPoints_EvolvedAFPArtifacts_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
ImplementationPoints_EvolvedAFPArtifacts

To :
ArtifactEffortComplexity
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightUpdatedTransactionalFunctionPoints

ID :
AutomatedEnhancementFunctionPoint_to_weightUpdatedTransactionalFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :
weightUpdatedTransactionalFunctionPoints
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightAddedTransactionalFunctionPoints

ID :
AutomatedEnhancementFunctionPoint_to_weightAddedTransactionalFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :
weightAddedTransactionalFunctionPoints
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightDeletedTransactionalFunctionPoints

ID :
AutomatedEnhancementFunctionPoint_to_weightDeletedTransactionalFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :
weightDeletedTransactionalFunctionPoints
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightUpdatedDataFunctionPoints

ID :
AutomatedEnhancementFunctionPoint_to_weightUpdatedDataFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :

weightUpdatedDataFunctionPoints
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightAddedDataFunctionPoints

ID :
AutomatedEnhancementFunctionPoint_to_weightAddedDataFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :
weightAddedDataFunctionPoints
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightDeletedDataFunctionPoints

ID :
AutomatedEnhancementFunctionPoint_to_weightDeletedDataFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :
weightDeletedDataFunctionPoints
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightSplitDataFunctionPoints

ID :
AutomatedEnhancementFunctionPoint_to_weightSplitDataFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :
weightSplitDataFunctionPoints
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightMergedDataFunctionPoints

ID :
AutomatedEnhancementFunctionPoint_to_weightMergedDataFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :
weightMergedDataFunctionPoints
-

MeasureRelationship AutomatedEnhancementFunctionPoint_to_weightDataFunctionPointsWithChangedType

ID :
AutomatedEnhancementFunctionPoint_to_weightDataFunctionPointsWithChangedType
Type :
BaseMeasureRelationship
From :
AutomatedEnhancementFunctionPoint
To :
weightDataFunctionPointsWithChangedType
-

MeasureRelationship weightDataFunctionPointsWithChangedType_to_DataAFPWithChangedTypeComplexityFactor

ID :
weightDataFunctionPointsWithChangedType_to_DataAFPWithChangedTypeComplexityFactor
Type :
Base1MeasureRelationship
From :
weightDataFunctionPointsWithChangedType
To :
DataAFPWithChangedTypeComplexityFactor
-

MeasureRelationship weightDataFunctionPointsWithChangedType_to_weightAutomatedFunctionPoints

ID :
weightDataFunctionPointsWithChangedType_to_weightAutomatedFunctionPoints
Type :
Base2MeasureRelationship
From :
weightDataFunctionPointsWithChangedType
To :
weightAutomatedFunctionPoints
-

MeasureRelationship weightMergedDataFunctionPoints_to_MergedDataAFPComplexityFactor

ID :
weightMergedDataFunctionPoints_to_MergedDataAFPComplexityFactor
Type :
Base1MeasureRelationship
From :
weightMergedDataFunctionPoints
To :
MergedDataAFPComplexityFactor
-

MeasureRelationship weightMergedDataFunctionPoints_to_weightAutomatedFunctionPoints

ID :
weightMergedDataFunctionPoints_to_weightAutomatedFunctionPoints
Type :
Base2MeasureRelationship
From :
weightMergedDataFunctionPoints
To :
weightAutomatedFunctionPoints
-

MeasureRelationship weightSplitDataFunctionPoints_to_SplitDataAFPComplexityFactor

ID :
weightSplitDataFunctionPoints_to_SplitDataAFPComplexityFactor
Type :
Base1MeasureRelationship
From :
weightSplitDataFunctionPoints
To :
SplitDataAFPComplexityFactor
-

MeasureRelationship weightSplitDataFunctionPoints_to_weightAutomatedFunctionPoints

ID :
weightSplitDataFunctionPoints_to_weightAutomatedFunctionPoints
Type :
Base2MeasureRelationship
From :
weightSplitDataFunctionPoints
To :
weightAutomatedFunctionPoints
-

MeasureRelationship weightUpdatedDataFunctionPoints_to_UpdatedDataAFPComplexityFactor

ID :
weightUpdatedDataFunctionPoints_to_UpdatedDataAFPComplexityFactor
Type :
Base1MeasureRelationship
From :
weightUpdatedDataFunctionPoints
To :
UpdatedDataAFPComplexityFactor
-

MeasureRelationship weightUpdatedDataFunctionPoints_to_weightAutomatedFunctionPoints

ID :
weightUpdatedDataFunctionPoints_to_weightAutomatedFunctionPoints
Type :

Base2MeasureRelationship
From :
weightUpdatedDataFunctionPoints
To :
weightAutomatedFunctionPoints
-

MeasureRelationship weightDeletedDataFunctionPoints_to_DeletedDataAFPComplexityFactor

ID :
weightDeletedDataFunctionPoints_to_DeletedDataAFPComplexityFactor
Type :
Base1MeasureRelationship
From :
weightDeletedDataFunctionPoints
To :
DeletedDataAFPComplexityFactor
-

MeasureRelationship weightDeletedDataFunctionPoints_to_weightAutomatedFunctionPoints

ID :
weightDeletedDataFunctionPoints_to_weightAutomatedFunctionPoints
Type :
Base2MeasureRelationship
From :
weightDeletedDataFunctionPoints
To :
weightAutomatedFunctionPoints
-

MeasureRelationship weightAddedDataFunctionPoints_to_AddedDataAFPComplexityFactor

ID :
weightAddedDataFunctionPoints_to_AddedDataAFPComplexityFactor
Type :
Base1MeasureRelationship
From :
weightAddedDataFunctionPoints
To :
AddedDataAFPComplexityFactor
-

MeasureRelationship weightAddedDataFunctionPoints_to_weightAutomatedFunctionPoints

ID :
weightAddedDataFunctionPoints_to_weightAutomatedFunctionPoints
Type :
Base2MeasureRelationship
From :
weightAddedDataFunctionPoints
To :

weightAutomatedFunctionPoints
-

MeasureRelationship UpdatedDataAFPComplexityFactor_to_DETChangeRatio

ID :
UpdatedDataAFPComplexityFactor_to_DETChangeRatio
Type :
RescaledMeasureRelationship
From :
UpdatedDataAFPComplexityFactor
To :
DETChangeRatio
-

MeasureRelationship DETChangeRatio_to_EvolvedDETInLatest

ID :
DETChangeRatio_to_EvolvedDETInLatest
Type :
Base1MeasureRelationship
From :
DETChangeRatio
To :
EvolvedDETInLatest
-

MeasureRelationship DETChangeRatio_to_TotalDETInPrevious

ID :
DETChangeRatio_to_TotalDETInPrevious
Type :
Base2MeasureRelationship
From :
DETChangeRatio
To :
TotalDETInPrevious
-

MeasureRelationship EvolvedDETInLatest_to_AddedDETInLatest

ID :
EvolvedDETInLatest_to_AddedDETInLatest
Type :
BaseMeasureRelationship
From :
EvolvedDETInLatest
To :
AddedDETInLatest
-

MeasureRelationship EvolvedDETInLatest_to_UpdatedDETInLatest

ID :

EvolvedDETInLatest_to_UpdatedDETInLatest
Type :
BaseMeasureRelationship
From :
EvolvedDETInLatest
To :
UpdatedDETInLatest
-

MeasureRelationship EvolvedDETInLatest_to_DeletedDETInLatest

ID :
EvolvedDETInLatest_to_DeletedDETInLatest
Type :
BaseMeasureRelationship
From :
EvolvedDETInLatest
To :
DeletedDETInLatest
-

MeasureRelationship weightUpdatedTransactionalFunctionPoints_to_UpdatedTransactionalAFPComplexityFactor

ID :
weightUpdatedTransactionalFunctionPoints_to_UpdatedTransactionalAFPComplexityFactor
Type :
Base1MeasureRelationship
From :
weightUpdatedTransactionalFunctionPoints
To :
UpdatedTransactionalAFPComplexityFactor
-

MeasureRelationship weightUpdatedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints

ID :
weightUpdatedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints
Type :
Base2MeasureRelationship
From :
weightUpdatedTransactionalFunctionPoints
To :
weightAutomatedFunctionPoints
-

MeasureRelationship weightDeletedTransactionalFunctionPoints_to_DeletedTransactionalAFPComplexityFactor

ID :
weightDeletedTransactionalFunctionPoints_to_DeletedTransactionalAFPComplexityFactor
Type :

Base1MeasureRelationship
From :
weightDeletedTransactionalFunctionPoints
To :
DeletedTransactionalAFPComplexityFactor
-

MeasureRelationship weightDeletedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints

ID :
weightDeletedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints
Type :
Base2MeasureRelationship
From :
weightDeletedTransactionalFunctionPoints
To :
weightAutomatedFunctionPoints
-

MeasureRelationship weightAddedTransactionalFunctionPoints_to_AddedTransactionalAFPComplexityFactor

ID :
weightAddedTransactionalFunctionPoints_to_AddedTransactionalAFPComplexityFactor
Type :
Base1MeasureRelationship
From :
weightAddedTransactionalFunctionPoints
To :
AddedTransactionalAFPComplexityFactor
-

MeasureRelationship weightAddedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints

ID :
weightAddedTransactionalFunctionPoints_to_weightAutomatedFunctionPoints
Type :
Base2MeasureRelationship
From :
weightAddedTransactionalFunctionPoints
To :
weightAutomatedFunctionPoints
-

MeasureRelationship UpdatedTransactionalAFPComplexityFactor_to_RawUpdatedTransactionalAFPComplexityFactor

ID :
UpdatedTransactionalAFPComplexityFactor_to_RawUpdatedTransactionalAFPComplexityFactor
Type :
BaseMeasureRelationship
From :

UpdatedTransactionalAFPComplexityFactor
To :
RawUpdatedTransactionalAFPComplexityFactor
-

MeasureRelationship UpdatedTransactionalAFPComplexityFactor_to_capRatioSharedEffortComplexity

ID :
UpdatedTransactionalAFPComplexityFactor_to_capRatioSharedEffortComplexity
Type :
BaseMeasureRelationship
From :
UpdatedTransactionalAFPComplexityFactor
To :
capRatioSharedEffortComplexity
-

MeasureRelationship sRatioEffortComplexity_to_RawUpdatedTransactionalAFPComplexityFactor

ID :
sRatioEffortComplexity_to_RawUpdatedTransactionalAFPComplexityFactor
Type :
RescaledMeasureRelationship
From :
sRatioEffortComplexity
To :
RawUpdatedTransactionalAFPComplexityFactor
-

MeasureRelationship sRatioEffortComplexity_to_wRatioEffortComplexityProcessed

ID :
sRatioEffortComplexity_to_wRatioEffortComplexityProcessed
Type :
BaseMeasureRelationship
From :
sRatioEffortComplexity
To :
wRatioEffortComplexityProcessed
-

MeasureRelationship sRatioEffortComplexity_to_wRatioEffortComplexityTotalVariation

ID :
sRatioEffortComplexity_to_wRatioEffortComplexityTotalVariation
Type :
BaseMeasureRelationship
From :
sRatioEffortComplexity
To :
wRatioEffortComplexityTotalVariation
-

MeasureRelationship RatioEffortComplexityProcessed_to_wRatioEffortComplexityProcessed

ID :
RatioEffortComplexityProcessed_to_wRatioEffortComplexityProcessed
Type :
RescaledMeasureRelationship
From :
RatioEffortComplexityProcessed
To :
wRatioEffortComplexityProcessed
-

MeasureRelationship RatioEffortComplexityTotalVariation_to_wRatioEffortComplexityTotalVariation

ID :
RatioEffortComplexityTotalVariation_to_wRatioEffortComplexityTotalVariation
Type :
RescaledMeasureRelationship
From :
RatioEffortComplexityTotalVariation
To :
wRatioEffortComplexityTotalVariation
-

MeasureRelationship RatioSharedEffortComplexity_to_capRatioSharedEffortComplexity

ID :
RatioSharedEffortComplexity_to_capRatioSharedEffortComplexity
Type :
RescaledMeasureRelationship
From :
RatioSharedEffortComplexity
To :
capRatioSharedEffortComplexity
-

MeasureRelationship RatioSharedEffortComplexity_to_SharedEffortComplexityProcessed

ID :
RatioSharedEffortComplexity_to_SharedEffortComplexityProcessed
Type :
Base1MeasureRelationship
From :
RatioSharedEffortComplexity
To :
SharedEffortComplexityProcessed
-

MeasureRelationship RatioSharedEffortComplexity_to_EffortComplexityProcessed

ID :
RatioSharedEffortComplexity_to_EffortComplexityProcessed
Type :

Base2MeasureRelationship
From :
RatioSharedEffortComplexity
To :
EffortComplexityProcessed
-

MeasureRelationship SharedEffortComplexityProcessed_to_SharedEffortComplexityAdded

ID :
SharedEffortComplexityProcessed_to_SharedEffortComplexityAdded
Type :
BaseMeasureRelationship
From :
SharedEffortComplexityProcessed
To :
SharedEffortComplexityAdded
-

MeasureRelationship SharedEffortComplexityProcessed_to_SharedEffortComplexityUpdated

ID :
SharedEffortComplexityProcessed_to_SharedEffortComplexityUpdated
Type :
BaseMeasureRelationship
From :
SharedEffortComplexityProcessed
To :
SharedEffortComplexityUpdated
-

MeasureRelationship SharedEffortComplexityProcessed_to_SharedEffortComplexityDeleted

ID :
SharedEffortComplexityProcessed_to_SharedEffortComplexityDeleted
Type :
BaseMeasureRelationship
From :
SharedEffortComplexityProcessed
To :
SharedEffortComplexityDeleted
-

MeasureRelationship SharedEffortComplexityDeleted_to_ArtifactEffortComplexity

ID :
SharedEffortComplexityDeleted_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
SharedEffortComplexityDeleted
To :

ArtifactEffortComplexity
-

MeasureRelationship SharedEffortComplexityUpdated_to_ArtifactEffortComplexity

ID :
SharedEffortComplexityUpdated_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
SharedEffortComplexityUpdated
To :
ArtifactEffortComplexity
-

MeasureRelationship SharedEffortComplexityAdded_to_ArtifactEffortComplexity

ID :
SharedEffortComplexityAdded_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
SharedEffortComplexityAdded
To :
ArtifactEffortComplexity
-

MeasureRelationship RatioEffortComplexityTotalVariation_to_EffortComplexityTotalVariation

ID :
RatioEffortComplexityTotalVariation_to_EffortComplexityTotalVariation
Type :
Base1MeasureRelationship
From :
RatioEffortComplexityTotalVariation
To :
EffortComplexityTotalVariation
-

MeasureRelationship RatioEffortComplexityTotalVariation_to_EffortComplexityTotalInPrevious

ID :
RatioEffortComplexityTotalVariation_to_EffortComplexityTotalInPrevious
Type :
Base2MeasureRelationship
From :
RatioEffortComplexityTotalVariation
To :
EffortComplexityTotalInPrevious
-

MeasureRelationship EffortComplexityTotalVariation_to_EffortComplexityTotalInLatest

ID :
EffortComplexityTotalVariation_to_EffortComplexityTotalInLatest
Type :
Base1MeasureRelationship
From :
EffortComplexityTotalVariation
To :
EffortComplexityTotalInLatest
-

MeasureRelationship EffortComplexityTotalVariation_to_EffortComplexityTotalInPrevious

ID :
EffortComplexityTotalVariation_to_EffortComplexityTotalInPrevious
Type :
Base2MeasureRelationship
From :
EffortComplexityTotalVariation
To :
EffortComplexityTotalInPrevious
-

MeasureRelationship EffortComplexityTotalInLatest_to_ArtifactEffortComplexity

ID :
EffortComplexityTotalInLatest_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
EffortComplexityTotalInLatest
To :
ArtifactEffortComplexity
-

MeasureRelationship RatioEffortComplexityProcessed_to_EffortComplexityProcessed

ID :
RatioEffortComplexityProcessed_to_EffortComplexityProcessed
Type :
Base1MeasureRelationship
From :
RatioEffortComplexityProcessed
To :
EffortComplexityProcessed
-

MeasureRelationship RatioEffortComplexityProcessed_to_EffortComplexityTotalInPrevious

ID :
RatioEffortComplexityProcessed_to_EffortComplexityTotalInPrevious
Type :

Base2MeasureRelationship
From :
RatioEffortComplexityProcessed
To :
EffortComplexityTotalInPrevious
-

MeasureRelationship EffortComplexityProcessed_to_EffortComplexityAdded

ID :
EffortComplexityProcessed_to_EffortComplexityAdded
Type :
BaseMeasureRelationship
From :
EffortComplexityProcessed
To :
EffortComplexityAdded
-

MeasureRelationship EffortComplexityProcessed_to_EffortComplexityUpdated

ID :
EffortComplexityProcessed_to_EffortComplexityUpdated
Type :
BaseMeasureRelationship
From :
EffortComplexityProcessed
To :
EffortComplexityUpdated
-

MeasureRelationship EffortComplexityProcessed_to_EffortComplexityDeleted

ID :
EffortComplexityProcessed_to_EffortComplexityDeleted
Type :
BaseMeasureRelationship
From :
EffortComplexityProcessed
To :
EffortComplexityDeleted
-

MeasureRelationship EffortComplexityTotalInPrevious_to_ArtifactEffortComplexity

ID :
EffortComplexityTotalInPrevious_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
EffortComplexityTotalInPrevious
To :

ArtifactEffortComplexity
-

MeasureRelationship EffortComplexityDeleted_to_ArtifactEffortComplexity

ID :
EffortComplexityDeleted_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
EffortComplexityDeleted
To :
ArtifactEffortComplexity
-

MeasureRelationship EffortComplexityUpdated_to_ArtifactEffortComplexity

ID :
EffortComplexityUpdated_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
EffortComplexityUpdated
To :
ArtifactEffortComplexity
-

MeasureRelationship EffortComplexityAdded_to_ArtifactEffortComplexity

ID :
EffortComplexityAdded_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
EffortComplexityAdded
To :
ArtifactEffortComplexity
-

MeasureRelationship EquivalenceRatio_Latest_to_AutomatedFunctionPoints_Latest

ID :
EquivalenceRatio_Latest_to_AutomatedFunctionPoints_Latest
Type :
Base1MeasureRelationship
From :
EquivalenceRatio_Latest
To :
AutomatedFunctionPoints_Latest
-

MeasureRelationship EquivalenceRatio_Latest_to_ImplementationPoints_LatestAFPScope

ID :
EquivalenceRatio_Latest_to_ImplementationPoints_LatestAFPScope
Type :
Base2MeasureRelationship
From :
EquivalenceRatio_Latest
To :
ImplementationPoints_LatestAFPScope
-

MeasureRelationship AutomatedFunctionPoints_Previous_to_weightAutomatedFunctionPoints

ID :
AutomatedFunctionPoints_Previous_to_weightAutomatedFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedFunctionPoints_Previous
To :
weightAutomatedFunctionPoints
-

MeasureRelationship AutomatedFunctionPoints_Latest_to_weightAutomatedFunctionPoints

ID :
AutomatedFunctionPoints_Latest_to_weightAutomatedFunctionPoints
Type :
BaseMeasureRelationship
From :
AutomatedFunctionPoints_Latest
To :
weightAutomatedFunctionPoints
-

MeasureRelationship ImplementationPoints_Previous_to_ImplementationPoints_PreviousAFPScope

ID :
ImplementationPoints_Previous_to_ImplementationPoints_PreviousAFPScope
Type :
BaseMeasureRelationship
From :
ImplementationPoints_Previous
To :
ImplementationPoints_PreviousAFPScope
-

MeasureRelationship ImplementationPoints_Previous_to_ImplementationPoints_PreviousATPScope

ID :
ImplementationPoints_Previous_to_ImplementationPoints_PreviousATPScope
Type :

BaseMeasureRelationship
From :
ImplementationPoints_Previous
To :
ImplementationPoints_PreviousATPScope
-

MeasureRelationship ImplementationPoints_Latest_to_ImplementationPoints_LatestAFPScope

ID :
ImplementationPoints_Latest_to_ImplementationPoints_LatestAFPScope
Type :
BaseMeasureRelationship
From :
ImplementationPoints_Latest
To :
ImplementationPoints_LatestAFPScope
-

MeasureRelationship ImplementationPoints_Latest_to_ImplementationPoints_LatestATPScope

ID :
ImplementationPoints_Latest_to_ImplementationPoints_LatestATPScope
Type :
BaseMeasureRelationship
From :
ImplementationPoints_Latest
To :
ImplementationPoints_LatestATPScope
-

MeasureRelationship ImplementationPoints_PreviousATPScope_to_ArtifactEffortComplexity

ID :
ImplementationPoints_PreviousATPScope_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
ImplementationPoints_PreviousATPScope
To :
ArtifactEffortComplexity
-

MeasureRelationship ImplementationPoints_LatestATPScope_to_ArtifactEffortComplexity

ID :
ImplementationPoints_LatestATPScope_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
ImplementationPoints_LatestATPScope
To :

ArtifactEffortComplexity
-

MeasureRelationship ImplementationPoints_PreviousAFPScope_to_ArtifactEffortComplexity

ID :
ImplementationPoints_PreviousAFPScope_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
ImplementationPoints_PreviousAFPScope
To :
ArtifactEffortComplexity
-

MeasureRelationship ImplementationPoints_LatestAFPScope_to_ArtifactEffortComplexity

ID :
ImplementationPoints_LatestAFPScope_to_ArtifactEffortComplexity
Type :
BaseMeasureRelationship
From :
ImplementationPoints_LatestAFPScope
To :
ArtifactEffortComplexity
-

MeasureRelationship ArtifactEffortComplexityLevel_to_ArtifactEffortComplexity

ID :
ArtifactEffortComplexityLevel_to_ArtifactEffortComplexity
Type :
RescaledMeasureRelationship
From :
ArtifactEffortComplexityLevel
To :
ArtifactEffortComplexity
-

MeasureRelationship ArtifactEffortComplexityValue_to_ArtifactEffortComplexityIndex

ID :
ArtifactEffortComplexityValue_to_ArtifactEffortComplexityIndex
Type :
RescaledMeasureRelationship
From :
ArtifactEffortComplexityValue
To :
ArtifactEffortComplexityIndex
-

MeasureRelationship ArtifactEffortComplexityValue_to_wArtifactCyclomaticComplexityLevel

ID :
ArtifactEffortComplexityValue_to_wArtifactCyclomaticComplexityLevel
Type :
BaseMeasureRelationship
From :
ArtifactEffortComplexityValue
To :
wArtifactCyclomaticComplexityLevel
-

MeasureRelationship ArtifactEffortComplexityValue_to_wArtifactSQLComplexityLevel

ID :
ArtifactEffortComplexityValue_to_wArtifactSQLComplexityLevel
Type :
BaseMeasureRelationship
From :
ArtifactEffortComplexityValue
To :
wArtifactSQLComplexityLevel
-

MeasureRelationship ArtifactEffortComplexityValue_to_wArtifactCouplingLevel

ID :
ArtifactEffortComplexityValue_to_wArtifactCouplingLevel
Type :
BaseMeasureRelationship
From :
ArtifactEffortComplexityValue
To :
wArtifactCouplingLevel
-

MeasureRelationship ArtifactEffortComplexityValue_to_wArtifactLinesOfCodeLevel

ID :
ArtifactEffortComplexityValue_to_wArtifactLinesOfCodeLevel
Type :
BaseMeasureRelationship
From :
ArtifactEffortComplexityValue
To :
wArtifactLinesOfCodeLevel
-

MeasureRelationship ArtifactEffortComplexityValue_to_wArtifactLackOfCommentLevel

ID :
ArtifactEffortComplexityValue_to_wArtifactLackOfCommentLevel
Type :

BaseMeasureRelationship
From :
ArtifactEffortComplexityValue
To :
wArtifactLackOfCommentLevel
-

MeasureRelationship ArtifactLackOfCommentLevel_to_wArtifactLackOfCommentLevel

ID :
ArtifactLackOfCommentLevel_to_wArtifactLackOfCommentLevel
Type :
RescaledMeasureRelationship
From :
ArtifactLackOfCommentLevel
To :
wArtifactLackOfCommentLevel
-

MeasureRelationship ArtifactLinesOfCodeLevel_to_wArtifactLinesOfCodeLevel

ID :
ArtifactLinesOfCodeLevel_to_wArtifactLinesOfCodeLevel
Type :
RescaledMeasureRelationship
From :
ArtifactLinesOfCodeLevel
To :
wArtifactLinesOfCodeLevel
-

MeasureRelationship ArtifactCouplingLevel_to_wArtifactCouplingLevel

ID :
ArtifactCouplingLevel_to_wArtifactCouplingLevel
Type :
RescaledMeasureRelationship
From :
ArtifactCouplingLevel
To :
wArtifactCouplingLevel
-

MeasureRelationship ArtifactSQLComplexityLevel_to_wArtifactSQLComplexityLevel

ID :
ArtifactSQLComplexityLevel_to_wArtifactSQLComplexityLevel
Type :
RescaledMeasureRelationship
From :
ArtifactSQLComplexityLevel
To :

wArtifactSQLComplexityLevel
-

MeasureRelationship ArtifactCyclomaticComplexityLevel_to_wArtifactCyclomaticComplexityLevel

ID :
ArtifactCyclomaticComplexityLevel_to_wArtifactCyclomaticComplexityLevel
Type :
RescaledMeasureRelationship
From :
ArtifactCyclomaticComplexityLevel
To :
wArtifactCyclomaticComplexityLevel
-

MeasureRelationship ArtifactCyclomaticComplexityLevel_to_ArtifactCyclomaticComplexity

ID :
ArtifactCyclomaticComplexityLevel_to_ArtifactCyclomaticComplexity
Type :
RankingMeasureRelationship
From :
ArtifactCyclomaticComplexityLevel
To :
ArtifactCyclomaticComplexity
-

MeasureRelationship ArtifactCouplingLevel_to_ArtifactFanIn

ID :
ArtifactCouplingLevel_to_ArtifactFanIn
Type :
RankingMeasureRelationship
From :
ArtifactCouplingLevel
To :
ArtifactFanIn
-

MeasureRelationship ArtifactLackOfCommentLevel_to_ArtifactCommentRatio

ID :
ArtifactLackOfCommentLevel_to_ArtifactCommentRatio
Type :
RankingMeasureRelationship
From :
ArtifactLackOfCommentLevel
To :
ArtifactCommentRatio
-

MeasureRelationship ArtifactLinesOfCodeLevel_to_ArtifactLinesOfCode

ID :
ArtifactLinesOfCodeLevel_to_ArtifactLinesOfCode
Type :
RankingMeasureRelationship
From :
ArtifactLinesOfCodeLevel
To :
ArtifactLinesOfCode
-

MeasureRelationship ArtifactSQLComplexityLevel_to_ArtifactSQLComplexity

ID :
ArtifactSQLComplexityLevel_to_ArtifactSQLComplexity
Type :
RankingMeasureRelationship
From :
ArtifactSQLComplexityLevel
To :
ArtifactSQLComplexity
-

MeasureRelationship ArtifactEffortComplexityLevel_to_ArtifactEffortComplexityIndex

ID :
ArtifactEffortComplexityLevel_to_ArtifactEffortComplexityIndex
Type :
RankingMeasureRelationship
From :
ArtifactEffortComplexityLevel
To :
ArtifactEffortComplexityIndex
-

MeasureRelationship ArtifactSQLComplexity_to_wArtifactMaxNumberOfSQLTablesPerQuery

ID :
ArtifactSQLComplexity_to_wArtifactMaxNumberOfSQLTablesPerQuery
Type :
BaseMeasureRelationship
From :
ArtifactSQLComplexity
To :
wArtifactMaxNumberOfSQLTablesPerQuery
-

MeasureRelationship ArtifactSQLComplexity_to_wArtifactNumberOfUsedSQLTables

ID :
ArtifactSQLComplexity_to_wArtifactNumberOfUsedSQLTables
Type :

BaseMeasureRelationship
From :
ArtifactSQLComplexity
To :
wArtifactNumberOfUsedSQLTables
-

MeasureRelationship ArtifactSQLComplexity_to_wArtifactNumberOfUsedSQLTableColumns

ID :
ArtifactSQLComplexity_to_wArtifactNumberOfUsedSQLTableColumns
Type :
BaseMeasureRelationship
From :
ArtifactSQLComplexity
To :
wArtifactNumberOfUsedSQLTableColumns
-

MeasureRelationship ArtifactSQLComplexity_to_wArtifactNumberOfSQLSubqueries

ID :
ArtifactSQLComplexity_to_wArtifactNumberOfSQLSubqueries
Type :
BaseMeasureRelationship
From :
ArtifactSQLComplexity
To :
wArtifactNumberOfSQLSubqueries
-

MeasureRelationship ArtifactSQLComplexity_to_wArtifactNumberOfGroupBySQLStatement

ID :
ArtifactSQLComplexity_to_wArtifactNumberOfGroupBySQLStatement
Type :
BaseMeasureRelationship
From :
ArtifactSQLComplexity
To :
wArtifactNumberOfGroupBySQLStatement
-

MeasureRelationship ArtifactSQLComplexity_to_wArtifactNumberOfUpdateSQLStatement

ID :
ArtifactSQLComplexity_to_wArtifactNumberOfUpdateSQLStatement
Type :
BaseMeasureRelationship
From :
ArtifactSQLComplexity
To :

wArtifactNumberOfUpdateSQLStatement
-

MeasureRelationship ArtifactNumberOfUpdateSQLStatement_to_wArtifactNumberOfUpdateSQLStatement

ID :
ArtifactNumberOfUpdateSQLStatement_to_wArtifactNumberOfUpdateSQLStatement
Type :
RescaledMeasureRelationship
From :
ArtifactNumberOfUpdateSQLStatement
To :
wArtifactNumberOfUpdateSQLStatement
-

MeasureRelationship ArtifactNumberOfGroupBySQLStatement_to_wArtifactNumberOfGroupBySQLStatement

ID :
ArtifactNumberOfGroupBySQLStatement_to_wArtifactNumberOfGroupBySQLStatement
Type :
RescaledMeasureRelationship
From :
ArtifactNumberOfGroupBySQLStatement
To :
wArtifactNumberOfGroupBySQLStatement
-

MeasureRelationship ArtifactNumberOfSQLSubqueries_to_wArtifactNumberOfSQLSubqueries

ID :
ArtifactNumberOfSQLSubqueries_to_wArtifactNumberOfSQLSubqueries
Type :
RescaledMeasureRelationship
From :
ArtifactNumberOfSQLSubqueries
To :
wArtifactNumberOfSQLSubqueries
-

MeasureRelationship ArtifactNumberOfUsedSQLTableColumns_to_wArtifactNumberOfUsedSQLTableColumns

ID :
ArtifactNumberOfUsedSQLTableColumns_to_wArtifactNumberOfUsedSQLTableColumns
Type :
RescaledMeasureRelationship
From :
ArtifactNumberOfUsedSQLTableColumns
To :
wArtifactNumberOfUsedSQLTableColumns

-

MeasureRelationship ArtifactNumberOfUsedSQLTables_to_wArtifactNumberOfUsedSQLTables

ID :
ArtifactNumberOfUsedSQLTables_to_wArtifactNumberOfUsedSQLTables
Type :
RescaledMeasureRelationship
From :
ArtifactNumberOfUsedSQLTables
To :
wArtifactNumberOfUsedSQLTables
-

MeasureRelationship ArtifactMaxNumberOfSQLTablesPerQuery_to_wArtifactMaxNumberOfSQLTablesPerQuery

ID :
ArtifactMaxNumberOfSQLTablesPerQuery_to_wArtifactMaxNumberOfSQLTablesPerQuery
Type :
RescaledMeasureRelationship
From :
ArtifactMaxNumberOfSQLTablesPerQuery
To :
wArtifactMaxNumberOfSQLTablesPerQuery
-

MeasureRelationship ArtifactMaxNumberOfSQLTablesPerQuery_to_QueryNumberOfSQLTables

ID :
ArtifactMaxNumberOfSQLTablesPerQuery_to_QueryNumberOfSQLTables
Type :
BaseMeasureRelationship
From :
ArtifactMaxNumberOfSQLTablesPerQuery
To :
QueryNumberOfSQLTables
-

MeasureRelationship AutomatedEnhancementPoint_to_AutomatedEnhancementFunctionPoint

ID :
AutomatedEnhancementPoint_to_AutomatedEnhancementFunctionPoint
Type :
Base1MeasureRelationship
From :
AutomatedEnhancementPoint
To :
AutomatedEnhancementFunctionPoint
-

MeasureRelationship AutomatedEnhancementPoint_to_AutomatedEnhancementTechnicalPoint

ID :
AutomatedEnhancementPoint_to_AutomatedEnhancementTechnicalPoint
Type :
Base2MeasureRelationship
From :
AutomatedEnhancementPoint
To :
AutomatedEnhancementTechnicalPoint
-

7.10 Instances of NamedMeasure Class

NamedMeasure weightExternalOutput

ID :
weightExternalOutput
Description :
Weight of a single External Output as defined by AFP 1.0 specifcations
Unit :
Cardinal
Scope :
kdm:Core::Element
Name :
afp:afp::wEO
Trait :
FeatureSizing
Category :
FunctionalMetrics
-

NamedMeasure weightExternalInput

ID :
weightExternalInput
Description :
Weight of a single External Input as defined by AFP 1.0 specifcations
Unit :
Cardinal
Scope :
kdm:Core::Element
Name :
afp:afp::wEI
Trait :
FeatureSizing
Category :
FunctionalMetrics

-

NamedMeasure weightExternalInterfaceFile

ID :
weightExternalInterfaceFile
Description :
Weight of a single External Interface File as defined by AFP 1.0 specifcations
Unit :
Cardinal
Scope :
kdm:Core::Element
Name :
afp:afp::wEIF
Trait :
FeatureSizing
Category :
FunctionalMetrics
-

NamedMeasure weightInternalLogicalFile

ID :
weightInternalLogicalFile
Description :
Weight of a single Internal Logical File as defined by AFP 1.0 specifcations
Unit :
Cardinal
Scope :
kdm:Core::Element
Name :
afp:afp::wILF
Trait :
FeatureSizing
Category :
FunctionalMetrics
-

NamedMeasure ArtifactLinesOfCode

ID :
ArtifactLinesOfCode
Description :
Number of Lines of Code (not Comment Code, nor blank)
Unit :
LineOfCode
Scope :
Artifact
Name :
smm:SMMSampleLibrary::CodeEltTotalLOC
Trait :

SizeComplexity
Category :
FunctionalMetrics
-

NamedMeasure ArtifactLinesOfCommentCode

ID :
ArtifactLinesOfCommentCode
Description :
Number of Lines of Comment Code
Unit :
Line
Scope :
Artifact
Name :
smm:SMMSampleLibrary::CodeEltTotalLOComment
Trait :
DocumentationComplexity
Category :
FunctionalMetrics
-

NamedMeasure ArtifactCyclomaticComplexity

ID :
ArtifactCyclomaticComplexity
Description :
Number of Edges in the Control Flow graph
Unit :
Edge
Scope :
Artifact
Name :
smm:SMMSampleLibrary::McCabeCyclomaticComplexity
Trait :
AlgorithmicComplexity
Category :
FunctionalMetrics
-

7.11 Instances of ObservationScope Class

ObservationScope FromRevision

ID :
FromRevision
Description :

Subset of the Application Model whith contains code elements from the previous / initial revision. Code elements are related to code elements from the latest / final revision by evolvedTo/evolvedFrom relationships.
-

ObservationScope ToRevision

ID :
ToRevision
Description :
Subset of the Application Model whith contains code elements from the latest / final revision. Code elements are related to code elements from the previous / initial revision by evolvedTo/evolvedFrom relationships.
-

7.12 Instances of OCLOperation Class

OCLOperations CallOrReferenceActions

ID :
CallOrReferenceActions
Context :
kdm:action::AbstractActionRelationship
Body :
((isOCLOfType(kdm:action::CallableRelations)
or isOCLOfType(kdm:action::DataRelations)
or isOCLOfType(kdm:code::Extends)
or isOCLOfType(kdm:code::Implements))
and from = self)
-

OCLOperations CalledOrReferencedCodeElements

ID :
CalledOrReferencedCodeElements
Context :
kdm:code::AbstractCodeElement
Body :
(self.CallOrReferenceActions.to())
-

OCLOperations CallAndReferenceGraph

ID :
CallAndReferenceGraph
Context :
kdm:code::AbstractCodeElement
Body :
(closure(CalledOrReferencedCodeElements()))
-

OCLOperations CallingOrReferencingActions

ID :
CallingOrReferencingActions
Context :
kdm:action::AbstractActionRelationship
Body :
((isOCLOfType(kdm:action::CallableRelations)
or isOCLOfType(kdm:action::DataRelations)
or isOCLOfType(kdm:code::Extends)
or isOCLOfType(kdm:code::Implements))
and to = self)
-

OCLOperations CallingOrReferencingCodeElements

ID :
CallingOrReferencingCodeElements
Context :
kdm:code::AbstractCodeElement
Body :
(self.CallingOrReferencingActions.from())
-

OCLOperations CallingAndReferencingGraph

ID :
CallingAndReferencingGraph
Context :
kdm:code::AbstractCodeElement
Body :
(closure(CallingOrReferencingCodeElements()))
-

OCLOperations AFPDataImplementationScope

ID :
AFPDataImplementationScope
Context :
DataAFP
Body :
(ownedElement())
-

OCLOperations AFPTransasctionImplementationScope

ID :
AFPTransasctionImplementationScope
Context :
AFPTransactionHead
Body :
(CallAndReferenceGraph())
-

OCLOperations AFPTransactionImplementationArtifacts

ID :
AFPTransactionImplementationArtifacts
Context :
AFPTransactionHead
Body :
(CallAndReferenceGraph()->select(a: Artifact))
-

OCLOperations isInLatestRevision

ID :
isInLatestRevision
Context :
Artifact
Body :
(ToRevision()->includes(self))
-

OCLOperations isInPreviousRevision

ID :
isInPreviousRevision
Context :
Artifact
Body :
(FromRevision()->includes(self))
-

OCLOperations isAddedArtifact

ID :
isAddedArtifact
Context :
Artifact
Body :
(isInLatestRevision
and not FromRevision()->exists(a: Artifacts
| a.evolvedTo = self))
-

OCLOperations isDeletedArtifact

ID :
isDeletedArtifact
Context :
Artifact
Body :
(isInPreviousRevision
and not ToRevision()->exists(a: Artifacts
| a.evolvedFrom = self))
-

OCLOperations isUpdatedArtifact

ID :
isUpdatedArtifact
Context :
Artifact
Body :
(isInLatestRevision
and ToRevision()->exists(a: Artifact
| a.evolvedTo = self and self.source <> a.source))
-

OCLOperations isUnchangedArtifact

ID :
isUnchangedArtifact
Context :
Artifact
Body :
(isInLatestRevision
and not (isUpdatedArtifact or isAddedArtifact))
-

OCLOperations isAddedAFPArtifact

ID :
isAddedAFPArtifact
Context :
Artifact
Body :
(isAddedArtifact and AFPScope()->includes(self))
-

OCLOperations isAddedSharedAFPArtifact

ID :
isAddedSharedAFPArtifact
Context :
Artifact
Body :
(isAddedArtifact and AFPSharedArtifacts()->includes(self))
-

OCLOperations isUpdatedAFPArtifact

ID :
isUpdatedAFPArtifact
Context :
Artifact
Body :
(isUpdatedArtifact and AFPScope()->includes(self))
-

OCLOperations isUpdatedSharedAFPArtifact

ID :
isUpdatedSharedAFPArtifact
Context :
Artifact
Body :
(isUpdatedArtifact and AFPSharedArtifacts()->includes(self))
-

OCLOperations isDeletedAFPArtifact

ID :
isDeletedAFPArtifact
Context :
Artifact
Body :
(isDeletedArtifact and AFPScope()->includes(self))
-

OCLOperations isDeletedSharedAFPArtifact

ID :
isDeletedSharedAFPArtifact
Context :
Artifact
Body :
(isDeletedArtifact and AFPSharedArtifacts()->includes(self))
-

OCLOperations isAddedDET

ID :
isAddedDET
Context :
DET
Body :
(isInLatestRevision
and not FromRevision()->exists(d: DET| d.evolvedTo = self))
-

OCLOperations isDeletedDET

ID :
isDeletedDET
Context :
DET
Body :
(isInPreviousRevision
and not ToRevision()->exists(d: DET | d.evolvedFrom = self))
-

OCLOperations isUpdatedDET

ID :
isUpdatedDET
Context :
DET
Body :
(isInLatestRevision
and ToRevision()->exists(d: DET
| d.evolvedTo = self and self.type <> d.type))
-

OCLOperations isUnchangedDET

ID :
isUnchangedDET
Context :
DET
Body :
(isInLatestRevision and not (isUpdatedDET or isAddedDET))
-

7.13 Instances of Operation Class

Operation Artifact_Recognizer

ID :
Artifact_Recognizer
Language :
OCL
Body :
((isOCLTypeOf(kdm:code::MethodUnit)
or (isOCLTypeOf(kdm:code:CallableUnit)
and (CallableKind = 'external'
or CallableKind = 'stored'
or CallableKind = 'regular')))
and name <> '')
-

Operation ILF_Recognizer

ID :
ILF_Recognizer
Language :
OCL
Body :
(afp:afp::isILF)
-

Operation EIF_Recognizer

ID :

EIF_Recognizer
Language :
OCL
Body :
(afp:afp::isEIF)
-

Operation TransactionalAFP_Recognizer

ID :
TransactionalAFP_Recognizer
Language :
OCL
Body :
(afp:afp::isEI or afp:afp::isEO)
-

Operation AFPTransactionHead_Recognizer

ID :
AFPTransactionHead_Recognizer
Language :
OCL
Body :
(EO().includes(self) or EI().includes(self))
-

Operation AFPSharedArtifacts_Recognizer

ID :
AFPSharedArtifacts_Recognizer
Language :
OCL
Body :
(CallingAndReferencingGraph()->exists(h1
| h1.isOCLOfType(AFPTransactionHead))
and CallingAndReferencingGraph()->exists(h2
| h2.isOCLOfType(AFPTransactionHead))
and h1 <> h2)
-

Operation AFPScope_Recognizer

ID :
AFPScope_Recognizer
Language :
OCL
Body :
(CallingAndReferencingGraph()->exists(h
| h.isOCLOfType(AFPTransactionHead)))
-

Operation ATPScope_Recognizer

ID :
ATPScope_Recognizer
Language :
OCL
Body :
(not AFPScope()->includes(self))
-

Operation LatestRevision_Recognizer

ID :
LatestRevision_Recognizer
Language :
OCL
Body :
(isInLatestRevision)
-

Operation LatestAFPScope_Recognizer

ID :
LatestAFPScope_Recognizer
Language :
OCL
Body :
(AFPScope()->includes(self)
and isInLatestRevision)
-

Operation LatestATPScope_Recognizer

ID :
LatestATPScope_Recognizer
Language :
OCL
Body :
(ATPScope()->includes(self)
and isInLatestRevision)
-

Operation PreviousRevision_Recognizer

ID :
PreviousRevision_Recognizer
Language :
OCL
Body :
(isInPreviousRevision)
-

Operation PreviousAFPScope_Recognizer

ID :
PreviousAFPScope_Recognizer
Language :
OCL
Body :
(AFPScope()->includes(self)
and isInPreviousRevision)
-

Operation PreviousATPScope_Recognizer

ID :
PreviousATPScope_Recognizer
Language :
OCL
Body :
(ATPScope()->includes(self)
and isInPreviousRevision)
-

Operation AddedTransactionalAFP_Recognizer

ID :
AddedTransactionalAFP_Recognizer
Language :
OCL
Body :
(isInLatestRevision
and not AFPTransactionHead()->exists(h
| h.isInPreviousRevision
and h.evolvedTo = self))
-

Operation DeletedTransactionalAFP_Recognizer

ID :
DeletedTransactionalAFP_Recognizer
Language :
OCL
Body :
(isInPreviousRevision
and not AFPTransactionHead()->exists(h
| h.isInLatestRevision
and h.evolvedFrom = self))
-

Operation UpdatedTransactionalAFP_Recognizer

ID :
UpdatedTransactionalAFP_Recognizer
Language :

OCL
Body :
(isInLatestRevision
and (AFPTransactionImplementationArtifacts()->exists(a
| a.isUpdatedArtifact)
or (AFPTransactionImplementationArtifacts() <>
evolvedFrom.AFPTransactionImplementationArtifacts()))
-

Operation AFPDataWithChangedType_Recognizer

ID :
AFPDataWithChangedType_Recognizer
Language :
OCL
Body :
(isInLatestRevision
and ((EIF()->includes(self)
and ILF()->exists(d
| d.isInPreviousRevision
and d.evolvedTo = self))
or (ILF()->includes(self)
and EIF()->exists(d
| d.isInPreviousRevision
and d.evolvedTo = self)))
-

Operation SplitDataAFP_Recognizer

ID :
SplitDataAFP_Recognizer
Language :
OCL
Body :
(d1, d2: DET
| d1 <> d2
and self.ownedElement().includes(d1)
and not self.ownedElement().includes(d2)
and not self.isInPreviousRevision
and DataAFP()->exists(l | l.isInPreviousRevision
and l.ownedElement().includes(d1.evolvedFrom)
and l.ownedElement(
-

Operation MergedDataAFP_Recognizer

ID :
MergedDataAFP_Recognizer
Language :
OCL
Body :

(ownedElement()->exists(d1, d2: DET
| not self.isInPreviousRevision
and self.ownedElement().includes(d2)
and self.ownedElement().includes(d1)
and DataAFP()-exists(l1, l2
| l1 <> l2
and l2.ownedElement().includes(d2.evolvedFrom)
and l1.ownedElement(
-

Operation AddedDataAFP_Recognizer

ID :
AddedDataAFP_Recognizer
Language :
OCL
Body :
(isInLatestRevision
and not DataAFP()->exists(d
| d.isInPreviousRevision
and d.evolvedTo = self))
-

Operation DeletedDataAFP_Recognizer

ID :
DeletedDataAFP_Recognizer
Language :
OCL
Body :
(isInPreviousRevision
and not DataAFP()->exists(d
| d.isInLatestRevision
and d.evolvedFrom = self))
-

Operation UpdatedDataAFP_Recognizer

ID :
UpdatedDataAFP_Recognizer
Language :
OCL
Body :
(isInLatestRevision
and (AFPDataImplementationScope()->exists(d: DET
| d.isUpdatedDET)
or (AFPDataImplementationScope() <> evolvedFrom.AFPDataImplementationScope()))
-

Operation EvolvedAFPArtifact_Recognizer

ID :

EvolvedAFPArtifact_Recognizer
Language :
OCL
Body :
(isAddedArtifact
or isUpdatedArtifact
or isDeletedArtifact)
-

Operation EvolvedATPArtifact_Recognizer

ID :
EvolvedATPArtifact_Recognizer
Language :
OCL
Body :
(isAddedArtifact
or isUpdatedArtifact
or isDeletedArtifact)
-

Operation DETInUpdatedDataAFP_Previous_Recognizer

ID :
DETInUpdatedDataAFP_Previous_Recognizer
Language :
OCL
Body :
(ownedElement()->select(d: DET
| d.isInPreviousRevision))
-

Operation DETInUpdatedDataAFP_Deleted_Recognizer

ID :
DETInUpdatedDataAFP_Deleted_Recognizer
Language :
OCL
Body :
(ownedElement()->select(d: DET
| d.isDeletedDET))
-

Operation DETInUpdatedDataAFP_Updated_Recognizer

ID :
DETInUpdatedDataAFP_Updated_Recognizer
Language :
OCL
Body :
(ownedElement()->select(d: DET
| d.isUpdatedDET))

-

Operation DETInUpdatedDataAFP_Added_Recognizer

ID :
DETInUpdatedDataAFP_Added_Recognizer
Language :
OCL
Body :
(ownedElement()->select(d: DET
| d.isAddedDET))
-

Operation SharedArtifactInUpdatedTransactionalAFP_Deleted_Recognizer

ID :
SharedArtifactInUpdatedTransactionalAFP_Deleted_Recognizer
Language :
OCL
Body :
(AFPTransactionImplementationArtifacts()->select(a
| a.isDeletedSharedAFPArtifact))
-

Operation SharedArtifactInUpdatedTransactionalAFP_Updated_Recognizer

ID :
SharedArtifactInUpdatedTransactionalAFP_Updated_Recognizer
Language :
OCL
Body :
(AFPTransactionImplementationArtifacts()->select(a
| a.isUpdatedSharedAFPArtifact))
-

Operation SharedArtifactInUpdatedTransactionalAFP_Added_Recognizer

ID :
SharedArtifactInUpdatedTransactionalAFP_Added_Recognizer
Language :
OCL
Body :
(AFPTransactionImplementationArtifacts()->select(a
| a.isAddedSharedAFPArtifact))
-

Operation ArtifactInUpdatedTransactionalAFP_Deleted_Recognizer

ID :
ArtifactInUpdatedTransactionalAFP_Deleted_Recognizer
Language :
OCL
Body :

(AFPTransactionImplementationArtifacts()->select(a
| a.isDeletedAFPArtifact))
-

Operation ArtifactInUpdatedTransactionalAFP_Updated_Recognizer

ID :
ArtifactInUpdatedTransactionalAFP_Updated_Recognizer
Language :
OCL
Body :
(AFPTransactionImplementationArtifacts()->select(a
| a.isUpdatedAFPArtifact))
-

Operation ArtifactInUpdatedTransactionalAFP_Added_Recognizer

ID :
ArtifactInUpdatedTransactionalAFP_Added_Recognizer
Language :
OCL
Body :
(AFPTransactionImplementationArtifacts()->select(a
| a.isAddedAFPArtifact))
-

Operation UpdatedTransactionalAFP_Latest_Recognizer

ID :
UpdatedTransactionalAFP_Latest_Recognizer
Language :
OCL
Body :
(AFPTransactionImplementationArtifacts()->select(a
| a.isInLatestRevision))
-

Operation UpdatedTransactionalAFP_Previous_Recognizer

ID :
UpdatedTransactionalAFP_Previous_Recognizer
Language :
OCL
Body :
(AFPTransactionImplementationArtifacts()->select(a
| a.isInPreviousRevision))
-

Operation ArtifactFanIn_Operation

ID :
ArtifactFanIn_Operation
Language :

OCL
Body :
(CallingAndReferencingActions()-size())
-

Operation ArtifactNumberOfUpdateSQLStatement_Operation

ID :
ArtifactNumberOfUpdateSQLStatement_Operation
Language :
OCL
Body :
(ownedElements()->select(w: WritesColumnSet)->size())
-

Operation ArtifactNumberOfGroupBySQLStatement_Operation

ID :
ArtifactNumberOfGroupBySQLStatement_Operation
Language :
OCL
Body :
(ownedElements()->select(g: kdm:Data::GroupContent)->size())
-

Operation ArtifactNumberOfSQLSubqueries_Operation

ID :
ArtifactNumberOfSQLSubqueries_Operation
Language :
OCL
Body :
(ownedElements()->select(a: kdm:Data::DataActions)->ownedElements()->select(sa: kdm:Data::DataActions)->size())
-

Operation ArtifactNumberOfUsedSQLTableColumns_Operation

ID :
ArtifactNumberOfUsedSQLTableColumns_Operation
Language :
OCL
Body :
(ownedElements()->select(c: kdm:Data::ColumnSet)->ownedElements()->select(i: kdm:Code::ItemUnit)->size())
-

Operation ArtifactNumberOfUsedSQLTables_Operation

ID :
ArtifactNumberOfUsedSQLTables_Operation
Language :
OCL

Body :
(ownedElements()->select(c: kdm:Data::ColumnSet)->size())
-

Operation QueryNumberOfSQLTables_Operation

ID :
QueryNumberOfSQLTables_Operation
Language :
OCL
Body :
(ownedElements()->select(c: kdm:Data::ColumnSet)->size())
-

Operation SQLQuery_Recognizer

ID :
SQLQuery_Recognizer
Language :
OCL
Body :
(isOCLOfType(kdm:Data::WritesColumnSet) or isOCLOfType(kdm:Data::ReadsColumnSet))
-

7.14 Instances of Ranking Class

Ranking ArtifactEffortComplexityLevel

ID :
ArtifactEffortComplexityLevel
Description :
Effort Complexity Level of an Artifact based on its Efforct Complexity Index
RankingMeasureRelationship :
ArtifactEffortComplexityLevel_to_ArtifactEffortComplexityIndex
Unit :
EffortComplexityLevel
Scope :
Artifact
Interval(s) :
LowEffortComplexity
ModerateEffortComplexity
HighEffortComplexity
VeryHighEffortComplexity
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

Ranking ArtifactSQLComplexityLevel

ID :

ArtifactSQLComplexityLevel
Description :
SQL Complexity level of an Artifact based on its overall SQL Complexity value
RankingMeasureRelationship :
ArtifactSQLComplexityLevel_to_ArtifactSQLComplexity
Unit :
SQLComplexityLevel
Scope :
Artifact
Interval(s) :
LowSQLComplexity
ModerateSQLComplexity
HighSQLComplexity
VeryHighSQLComplexity
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

Ranking ArtifactLinesOfCodeLevel

ID :
ArtifactLinesOfCodeLevel
Description :
Size level of an Artifact based on its number of Lines of Code
RankingMeasureRelationship :
ArtifactLinesOfCodeLevel_to_ArtifactLinesOfCode
Unit :
LinesOfCodeLevel
Scope :
Artifact
Interval(s) :
SmallLinesOfCode
MediumLinesOfCode
LargeLinesOfCode
VeryLargeLinesOfCode
Trait :
SizeComplexity
Category :
FunctionalMetrics
-

Ranking ArtifactLackOfCommentLevel

ID :
ArtifactLackOfCommentLevel
Description :
Lack of Comment level of an Artifact based on its number of Lines of Code and Lines of Comment Code
RankingMeasureRelationship :

ArtifactLackOfCommentLevel_to_ArtifactCommentRatio
Unit :
LackOfCommentLevel
Scope :
Artifact
Interval(s) :
LowLackOfComment
ModerateLackOfComment
HighLackOfComment
VeryHighLackOfComment
Trait :
DocumentationComplexity
Category :
FunctionalMetrics
-

Ranking ArtifactCouplingLevel

ID :
ArtifactCouplingLevel
Description :
Coupling level of an Artifact based on its Fan-In
RankingMeasureRelationship :
ArtifactCouplingLevel_to_ArtifactFanIn
Unit :
CouplingLevel
Scope :
Artifact
Interval(s) :
LowCoupling
ModerateCoupling
HighCoupling
VeryHighCoupling
Trait :
CouplingComplexity
Category :
FunctionalMetrics
-

Ranking ArtifactCyclomaticComplexityLevel

ID :
ArtifactCyclomaticComplexityLevel
Description :
Cyclomatic Complexity level of an Artifact based on its McCabe Cyclomatic Complexity value
RankingMeasureRelationship :
ArtifactCyclomaticComplexityLevel_to_ArtifactCyclomaticComplexity
Unit :
CyclomaticComplexityLevel
Scope :

Artifact
Interval(s) :
LowCyclomaticComplexity
ModerateCyclomaticComplexity
HighCyclomaticComplexity
VeryHighCyclomaticComplexity
Trait :
AlgorithmicComplexity
Category :
FunctionalMetrics
-

7.15 Instances of RankingInterval Class

RankingInterval LowEffortComplexity

ID :
LowEffortComplexity
Symbol :
LowEC
Minimum (if any : True) :
0
Maximum (if any : True) :
0
-

RankingInterval ModerateEffortComplexity

ID :
ModerateEffortComplexity
Symbol :
ModerateEC
Minimum (if any : True) :
1
Maximum (if any : True) :
1
-

RankingInterval HighEffortComplexity

ID :
HighEffortComplexity
Symbol :
HighEC
Minimum (if any : True) :
2
Maximum (if any : True) :
2
-

RankingInterval VeryHighEffortComplexity

ID :
VeryHighEffortComplexity
Symbol :
VeryHighEC
Minimum (if any : True) :
3

Maximum (if any : True) :
3
-

RankingInterval LowSQLComplexity

ID :
LowSQLComplexity
Symbol :
LowSQLComplexity
Minimum (if any : True) :
0
Maximum (if any : True) :
{ Integer maxModerateSQLComplexity = 10 }
-

RankingInterval ModerateSQLComplexity

ID :
ModerateSQLComplexity
Symbol :
ModerateSQLComplexity
Minimum (if any : True) :
{ Integer maxModerateSQLComplexity = 10 }
Maximum (if any : True) :
{ Integer maxHighSQLComplexity = 40 }
-

RankingInterval HighSQLComplexity

ID :
HighSQLComplexity
Symbol :
HighSQLComplexity
Minimum (if any : True) :
{ Integer maxHighSQLComplexity = 40 }
Maximum (if any : True) :
{ Integer maxVeryHighSQLComplexity = 70 }
-

RankingInterval VeryHighSQLComplexity

ID :
VeryHighSQLComplexity
Symbol :
VeryHighSQLComplexity
Minimum (if any : True) :
{ Integer maxVeryHighSQLComplexity = 70 }
Maximum (if any : True) :

100
-

RankingInterval SmallLinesOfCode

ID :
SmallLinesOfCode
Symbol :
SmallLOC
Minimum (if any : True) :
0
Maximum (if any : True) :
{ Integer maxMediumLOC = 10 }
-

RankingInterval MediumLinesOfCode

ID :
MediumLinesOfCode
Symbol :
MediumLOC
Minimum (if any : True) :
{ Integer maxMediumLOC = 10 }
Maximum (if any : True) :
{ Integer maxLargeLOC = 50 }
-

RankingInterval LargeLinesOfCode

ID :
LargeLinesOfCode
Symbol :
LargeLOC
Minimum (if any : True) :
{ Integer maxLargeLOC = 50 }
Maximum (if any : True) :
{ Integer maxVeryLargeLOC = 200 }
-

RankingInterval VeryLargeLinesOfCode

ID :
VeryLargeLinesOfCode
Symbol :
VeryLargeLOC
Minimum (if any : True) :
{ Integer maxVeryLargeLOC = 200 }
Maximum (if any : False) :

-

RankingInterval LowLackOfComment

ID :
LowLackOfComment
Symbol :
LowLackOfComment
Minimum (if any : True) :
{ Integer minModerateLackOfComment = 15 }
Maximum (if any : False) :
-

RankingInterval ModerateLackOfComment

ID :
ModerateLackOfComment
Symbol :
ModerateLackOfComment
Minimum (if any : True) :
{ Integer minHighLackOfComment = 7 }
Maximum (if any : True) :
{ Integer minModerateLackOfComment = 15 }
-

RankingInterval HighLackOfComment

ID :
HighLackOfComment
Symbol :
HighLackOfComment
Minimum (if any : True) :
{ Integer minVeryHighLackOfComment = 3 }
Maximum (if any : True) :
{ Integer minHighLackOfComment = 7 }
-

RankingInterval VeryHighLackOfComment

ID :
VeryHighLackOfComment
Symbol :
VeryHighLackOfComment
Minimum (if any : True) :
0
Maximum (if any : True) :
{ Integer minVeryHighLackOfComment = 3 }
-

RankingInterval LowCoupling

ID :
LowCoupling
Symbol :
LowCoupling
Minimum (if any : True) :
0
Maximum (if any : True) :
{ Integer maxModerateFI = 4 }
-

RankingInterval ModerateCoupling

ID :
ModerateCoupling
Symbol :
ModerateCoupling
Minimum (if any : True) :
{ Integer maxModerateFI = 4 }
Maximum (if any : True) :
{ Integer maxHighFI = 10 }
-

RankingInterval HighCoupling

ID :
HighCoupling
Symbol :
HighCoupling
Minimum (if any : True) :
{ Integer maxHighFI = 10 }
Maximum (if any : True) :
{ Integer maxVeryHighFI = 30 }
-

RankingInterval VeryHighCoupling

ID :
VeryHighCoupling
Symbol :
VeryHighCoupling
Minimum (if any : True) :
{ Integer maxVeryHighFI = 30 }
Maximum (if any : False) :
-

RankingInterval LowCyclomaticComplexity

ID :
LowCyclomaticComplexity
Symbol :
LowCC
Minimum (if any : True) :
1
Maximum (if any : True) :
{ Integer maxModerateCC = 5 }
-

RankingInterval ModerateCyclomaticComplexity

ID :
ModerateCyclomaticComplexity
Symbol :
ModerateCC
Minimum (if any : True) :
{ Integer maxModerateCC = 5 }
Maximum (if any : True) :
{ Integer maxHighCC = 15 }
-

RankingInterval HighCyclomaticComplexity

ID :
HighCyclomaticComplexity
Symbol :
HighCC
Minimum (if any : True) :
{ Integer maxHighCC = 15 }
Maximum (if any : True) :
{ Integer maxVeryHighCC = 30 }
-

RankingInterval VeryHighCyclomaticComplexity

ID :
VeryHighCyclomaticComplexity
Symbol :
VeryHighCC
Minimum (if any : True) :
{ Integer maxVeryHighCC = 30 }
Maximum (if any : False) :
-

7.16 Instances of RatioMeasure Class

RatioMeasure ArtifactCommentRatio

ID :
ArtifactCommentRatio
Description :
Percentage of Lines of Comment Code among Lines of Code and Lines of Comment Code, ranging from 0% to 100%
Functor :
divide
Unit :
Percentage
Scope :
Artifact
BaseMeasure1Relationship :
ArtifactCommentRatio_to_ArtifactLinesOfCommentCode
BaseMeasure2Relationship :
ArtifactCommentRatio_to_ArtifactLinesOfCodeAndCommentCode
Trait :
DocumentationComplexity
Category :
FunctionalMetrics
-

RatioMeasure DETChangeRatio

ID :
DETChangeRatio
Description :
Percentage of evolved DET in Updated Data AFP
Functor :
divide
Unit :
Percentage
Scope :
UpdatedDataAFP
BaseMeasure1Relationship :
DETChangeRatio_to_EvolvedDETInLatest
BaseMeasure2Relationship :
DETChangeRatio_to_TotalDETInPrevious
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

RatioMeasure RatioSharedEffortComplexity

ID :
RatioSharedEffortComplexity
Description :
Percentage of Effort Complexity in Shared evolved Artifacts of a Transactional AFP among the Effort Complexity in all evolved Artifacts
Functor :
divide
Unit :
Percentage
Scope :
UpdatedTransactionalAFP
BaseMeasure1Relationship :
RatioSharedEffortComplexity_to_SharedEffortComplexityProcessed
BaseMeasure2Relationship :
RatioSharedEffortComplexity_to_EffortComplexityProcessed
Trait :
FeatureEnhancementSizing
Category :
FunctionalMetrics
-

RatioMeasure RatioEffortComplexityTotalVariation

ID :
RatioEffortComplexityTotalVariation
Description :
Ratio of Effort Complexity net variation between latest / final revision and previous / initial revision on the Effort Complexity of Transactional AFP in previous / initial revision
Functor :
divide
Unit :
Percentage
Scope :
UpdatedTransactionalAFP
BaseMeasure1Relationship :
RatioEffortComplexityTotalVariation_to_EffortComplexityTotalVariation
BaseMeasure2Relationship :
RatioEffortComplexityTotalVariation_to_EffortComplexityTotalInPrevious
Trait :
FeatureEnhancementSizing
Category :
FunctionalMetrics
-

RatioMeasure RatioEffortComplexityProcessed

ID :
RatioEffortComplexityProcessed
Description :
Percentage of Effort Complexity in evolved Artifacts of a Transactional AFP among the Effort Complexity of all Artifacts
Functor :
divide
Unit :
Percentage
Scope :
UpdatedTransactionalAFP
BaseMeasure1Relationship :
RatioEffortComplexityProcessed_to_EffortComplexityProcessed
BaseMeasure2Relationship :
RatioEffortComplexityProcessed_to_EffortComplexityTotalInPrevious
Trait :
FeatureEnhancementSizing
Category :
FunctionalMetrics
-

RatioMeasure EquivalenceRatio_Latest

ID :
EquivalenceRatio_Latest
Description :
Average Automated Function Point value delivered by an Implementation Point, as measured in the latest / final revision
Functor :
divide
Unit :
AutomatedFunctionPointPerImplementationPoints
Scope :
LatestRevision
BaseMeasure1Relationship :
EquivalenceRatio_Latest_to_AutomatedFunctionPoints_Latest
BaseMeasure2Relationship :
EquivalenceRatio_Latest_to_ImplementationPoints_LatestAFPScope
Trait :
TechnicalEnhancementSizing
Category :
FunctionalMetrics
-

7.17 Instances of RescaledMeasure Class

RescaledMeasure UpdatedDataAFPComplexityFactor

ID :
UpdatedDataAFPComplexityFactor
Description :
Complexity Factor to use to weight Updated Data AFP to get the resulting Automated Enhancement Function Points, based on the DET Change Ratio
Unit :
Cardinal
Scope :
UpdatedDataAFP
RescaledMeasureRelationship :
UpdatedDataAFPComplexityFactor_to_DETChangeRatio
Formula :
IF DETChangeRatio <= 1/3 THEN 0.25; IF 1/3 < DETChangeRatio <= 2/3 THEN 0.50; IF 2/3 < DETChangeRatio <= 1 THEN 0.75; IF DETChangeRatio > 1 THEN 1;
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure DataAFPWithChangedTypeComplexityFactor

ID :
DataAFPWithChangedTypeComplexityFactor
Description :
Complexity Factor to use to weight Data AFP with Changed Type (EIF to ILF or ILF to EIF) to get the resulting Automated Enhancement Function Points
Unit :
Real
Scope :
DataAFPWithChangedType
RescaledMeasureRelationship :
Formula :
{ Real changedTypeCF = 0.40 };
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure MergedDataAFPComplexityFactor

ID :
MergedDataAFPComplexityFactor
Description :
Complexity Factor to use to weight Data AFP resulting from the merge of Data AFP from the previous / initial revision to get the resulting Automated Enhancement Function Points
Unit :
Real
Scope :
MergedDataAFP
RescaledMeasureRelationship :
Formula :
{ Real mergedCF = 0.40 };
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure SplitDataAFPComplexityFactor

ID :
SplitDataAFPComplexityFactor
Description :
Complexity Factor to use to weight Data AFP resulting from the split of a Data AFP in the previous / initial revision to get the resulting Automated Enhancement Function Points
Unit :
Real
Scope :
SplitDataAFP
RescaledMeasureRelationship :
Formula :
{ Real splitCF = 0.40 };
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure DeletedDataAFPComplexityFactor

ID :
DeletedDataAFPComplexityFactor
Description :
Complexity Factor to use to weight Deleted Data AFP from previous / initial revision to get the resulting Automated Enhancement Function Points

Unit :
Real
Scope :
DeletedDataAFP
RescaledMeasureRelationship :
Formula :
{ Real deletedCF = 0.40 };
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure AddedDataAFPComplexityFactor

ID :
AddedDataAFPComplexityFactor
Description :
Complexity Factor to use to weight Added Data AFP in latest / final revision to get the resulting Automated Enhancement Function Points
Unit :
Real
Scope :
AddedDataAFP
RescaledMeasureRelationship :
Formula :
{ Real addedCF = 1 };
Trait :
DataEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure DeletedTransactionalAFPComplexityFactor

ID :
DeletedTransactionalAFPComplexityFactor
Description :
Complexity Factor to use to weight Deleted Transactional AFP from previous / initial revision to get the resulting Automated Enhancement Function Points
Unit :
Real
Scope :
DeletedTransactionalAFP
RescaledMeasureRelationship :
Formula :

{ Real deletedCF = 0.40 };
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure AddedTransactionalAFPComplexityFactor

ID :
AddedTransactionalAFPComplexityFactor
Description :
Complexity Factor to use to weight Added Transactional AFP in latest / final revision to get the resulting Automated Enhancement Function Points
Unit :
Real
Scope :
AddedTransactionalAFP
RescaledMeasureRelationship :
Formula :
{ Real addedCF = 1 };
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure RawUpdatedTransactionalAFPComplexityFactor

ID :
RawUpdatedTransactionalAFPComplexityFactor
Description :
Raw Complexity Factor to use to weight Updated Transactional AFP to get the resulting Automated Enhancement Function Points, based on the Effort Complexity Net Variation Ratio and the Processed Effort Complexity Ration, before the capping based on the Rat
Unit :
Real
Scope :
UpdatedTransactionalAFP
RescaledMeasureRelationship :
sRatioEffortComplexity_to_RawUpdatedTransactionalAFPComplexityFactor
Formula :
IF sRatioEffortComplexity = 0 THEN 0.25;
IF sRatioEffortComplexity = 1 THEN 0.50;
IF sRatioEffortComplexity = 2 THEN 0.75;
IF sRatioEffortComplexity = 3 THEN 1.00;

IF sRatioEffortComplexity = 4 THEN 1.25;
IF sRatioEffortComplexity = 5 THEN 1.50
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure wRatioEffortComplexityProcessed

ID :
wRatioEffortComplexityProcessed
Description :
Weight of the Processed Effort Complexity Ratio in the Raw Complexity Factor of Updated Transactional AFP
Unit :
Cardinal
Scope :
UpdatedTransactionalAFP
RescaledMeasureRelationship :
RatioEffortComplexityProcessed_to_wRatioEffortComplexityProcessed
Formula :
IF RatioEffortComplexityProcessed <= 1/3 THEN 0;
IF 1/3 < RatioEffortComplexityProcessed <= 2/3 THEN 1;
IF 2/3 < RatioEffortComplexityProcessed <= 1 THEN 2;
IF RatioEffortComplexityProcessed > 1 THEN 3
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure wRatioEffortComplexityTotalVariation

ID :
wRatioEffortComplexityTotalVariation
Description :
Weight of the Effort Complexity Net Variation Ratio in the Raw Complexity Factor of Updated Transactional AFP
Unit :
Cardinal
Scope :
UpdatedTransactionalAFP
RescaledMeasureRelationship :
RatioEffortComplexityTotalVariation_to_wRatioEffortComplexityTotalVariation
Formula :

IF RatioEffortComplexityTotalVariation <= 1/3 THEN 0;
IF 1/3 < RatioEffortComplexityTotalVariation <= 2/3 THEN 1;
IF 2/3 < RatioEffortComplexityTotalVariation <= 1 THEN 2;
IF RatioEffortComplexityTotalVariation > 1 THEN 3;
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure capRatioSharedEffortComplexity

ID :
capRatioSharedEffortComplexity
Description :
Cap value for the Complexity Factor of Updated Transactional AFP based on the Shared Effort Complexity Ratio
Unit :
Real
Scope :
UpdatedTransactionalAFP
RescaledMeasureRelationship :
RatioSharedEffortComplexity_to_capRatioSharedEffortComplexity
Formula :
IF RatioShareEffortComplexity = 1 THEN 0.25;
IF 1 > RatioShareEffortComplexity >= 3/4 THEN 0.50;
IF 3/4 > RatioShareEffortComplexity >= 1/2 THEN 0.75;
IF 1/2 > RatioShareEffortComplexity THEN 2;
Trait :
TransactionalEvolutionSizing
Category :
FunctionalMetrics
-

RescaledMeasure ArtifactEffortComplexity

ID :
ArtifactEffortComplexity
Description :
Effort Complexity value of an Artifact based on its Effort Complexity Level
Unit :
ImplementationPoint
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactEffortComplexityLevel_to_ArtifactEffortComplexity

Formula :
IF VeryHighEC THEN { Real wVeryHighEC = 1.2 };
IF HighEC THEN { Real wHighEC = 0.7 };
IF ModerateEC THEN { Real wModerateEC = 0.2 };
IF LowEC THEN { Real wLowEC = 0.1 }
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

RescaledMeasure ArtifactEffortComplexityIndex

ID :
ArtifactEffortComplexityIndex
Description :
Decoding of contributing bits into an aggregated Effort Complexity Index ranging from 0 to 3
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactEffortComplexityValue_to_ArtifactEffortComplexityIndex
Formula :
IF (ArtifactEffortComplexityValue & 2^0 = 2^0 OR ArtifactEffortComplexityValue & 2^1 = 2^1 AND (ArtifactEffortComplexityValue & 2^2 = 2^2 AND ArtifactEffortComplexityValue & 2^3 = 2^3 OR ArtifactEffortComplexityValue & 2^4 = 2^4) THEN 3
ELSE IF (A
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactLackOfCommentLevel

ID :
wArtifactLackOfCommentLevel
Description :
Contribution bit of the Lack of Comment level into the overall Effort Complexity
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactLackOfCommentLevel_to_wArtifactLackOfCommentLevel

Formula :
IF VeryHighLackOfComment THEN 2^4;
IF HighLackOfComment THEN 2^9;
IF ModerateLackOfComment THEN 2^14;
IF LowLackOfComment THEN 2^19
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactLinesOfCodeLevel

ID :
wArtifactLinesOfCodeLevel
Description :
Contribution bit of the Lines Of Code level into the overall Effort Complexity
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactLinesOfCodeLevel_to_wArtifactLinesOfCodeLevel
Formula :
IF VeryLargeLinesOfCode THEN 2^3;
IF LargeLinesOfCode THEN 2^8;
IF MediumLinesOfCode THEN 2^13;
IF SmallLinesOfCode THEN 2^18
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactCouplingLevel

ID :
wArtifactCouplingLevel
Description :
Contribution bit of the inbound Coupling level into the overall Effort Complexity
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactCouplingLevel_to_wArtifactCouplingLevel

Formula :
IF VeryHighCoupling THEN 2^2;
IF HighCoupling THEN 2^7;
IF ModerateCoupling THEN 2^12;
IF LowCoupling THEN 2^17
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactSQLComplexityLevel

ID :
wArtifactSQLComplexityLevel
Description :
Contribution bit of the SQL Complexity level into the overall Effort Complexity
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactSQLComplexityLevel_to_wArtifactSQLComplexityLevel
Formula :
IF VeryHighSQLComplexity THEN 2;
IF HighSQLComplexity THEN 2^6;
IF ModerateSQLComplexity THEN 2^11;
IF LowSQLComplexity THEN 2^16
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactCyclomaticComplexityLevel

ID :
wArtifactCyclomaticComplexityLevel
Description :
Contribution bit of the Cyclomatic Complexity level into the overall Effort Complexity
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactCyclomaticComplexityLevel_to_wArtifactCyclomaticComplexityLevel

Formula :
IF VeryHighCyclomaticComplexity THEN 1;
IF HighCyclomaticComplexity THEN 2^5;
IF ModerateCyclomaticComplexity THEN 2^10;
IF LowCyclomaticComplexity THEN 2^15
Trait :
ImplementationComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactNumberOfUpdateSQLStatement

ID :
wArtifactNumberOfUpdateSQLStatement
Description :
Weight of presence of an Update SQL query into the overall SQL Complexity of an Artifact
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactNumberOfUpdateSQLStatement_to_wArtifactNumberOfUpdateSQLStatement
Formula :
IF ArtifactNumberOfUpdateSQLStatement > { Integer maxUpdateSQLStatement = 0 }
THEN { Integer wUpdateSQLStatement = 10 };
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactNumberOfGroupBySQLStatement

ID :
wArtifactNumberOfGroupBySQLStatement
Description :
Weight of presence of a SQL query with a Group By into the overall SQL Complexity of an Artifact
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactNumberOfGroupBySQLStatement_to_wArtifactNumberOfGroupBySQLStatement
Formula :
IF ArtifactNumberOfGroupBySQLStatement > { Integer maxGroupBySQLStatement = 0 }

THEN { Integer wGroupBySQLStatement = 0 };
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactNumberOfSQLSubqueries

ID :
wArtifactNumberOfSQLSubqueries
Description :
Weight of presence of a SQL subquery into the overall SQL Complexity of an Artifact
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactNumberOfSQLSubqueries_to_wArtifactNumberOfSQLSubqueries
Formula :
IF ArtifactNumberOfSQLSubqueries > { Integer maxSQLSubqueries = 0 }
THEN { Integer wSQLSubqueries = 10 };
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactNumberOfUsedSQLTableColumns

ID :
wArtifactNumberOfUsedSQLTableColumns
Description :
Weight of presence of a SQL query on too many columns into the overall SQL Complexity of an Artifact
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactNumberOfUsedSQLTableColumns_to_wArtifactNumberOfUsedSQLTableColumns
Formula :
IF ArtifactNumberOfUsedSQLTableColumns > { Integer maxSQLTableColumns = 9 }
THEN { Integer wSQLTableColumns = 10 };
Trait :
DataAccessComplexity
Category :

FunctionalMetrics
-

RescaledMeasure wArtifactNumberOfUsedSQLTables

ID :
wArtifactNumberOfUsedSQLTables
Description :
Weight of presence of SQL queries on too many tables into the overall SQL Complexity of an Artifact
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactNumberOfUsedSQLTables_to_wArtifactNumberOfUsedSQLTables
Formula :
IF ArtifactNumberOfUsedSQLTables > { Integer maxSQLTables = 30 }
THEN { Integer wSQLTables = 10 } ;
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

RescaledMeasure wArtifactMaxNumberOfSQLTablesPerQuery

ID :
wArtifactMaxNumberOfSQLTablesPerQuery
Description :
Weight of presence of a SQL query on too many tables into the overall SQL Complexity of an Artifact
Unit :
Cardinal
Scope :
Artifact
RescaledMeasureRelationship :
ArtifactMaxNumberOfSQLTablesPerQuery_to_wArtifactMaxNumberOfSQLTablesPerQuery
Formula :
IF ArtifactMaxNumberOfSQLTablesPerQuery > { Integer maxSQLTablesPerQuery = 4 }
THEN { Integer wSQLTablesPerQuery = 50 };
Trait :
DataAccessComplexity
Category :
FunctionalMetrics
-

7.18 Instances of Scope Class

Scope InventoryModel

ID :
InventoryModel
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
-

Scope Artifact

ID :
Artifact
Class :
kdm:code::ControlElement
Recognizer (if any) :
Artifact_Recognizer
-

Scope DataAFP

ID :
DataAFP
Class :
afp:afp::LF
Recognizer (if any) :
-

Scope DET

ID :
DET
Class :
afp:afp::Element
Recognizer (if any) :
-

Scope ILF

ID :
ILF
Class :
kdm:code::DataElement
Recognizer (if any) :
ILF_Recognizer
-

Scope EIF

ID :
EIF
Class :
kdm:code::DataElement
Recognizer (if any) :
EIF_Recognizer
-

Scope EI

ID :
EI
Class :
afp:afp::EI
Recognizer (if any) :
-

Scope EO

ID :
EO
Class :
afp:afp::EO
Recognizer (if any) :
-

Scope TransactionalAFP

ID :
TransactionalAFP
Class :
kdm:Core::Element
Recognizer (if any) :
TransactionalAFP_Recognizer
-

Scope AFPTransactionHead

ID :
AFPTransactionHead
Class :
kdm:ui::AbstractUIElement
Recognizer (if any) :
AFPTransactionHead_Recognizer
-

Scope AFPSharedArtifacts

ID :
AFPSharedArtifacts
Class :
Artifact
Recognizer (if any) :
AFPSharedArtifacts_Recognizer
-

Scope AFPScope

ID :
AFPScope
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
AFPScope_Recognizer
-

Scope ATPScope

ID :
ATPScope
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
ATPScope_Recognizer
-

Scope LatestRevision

ID :
LatestRevision
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
LatestRevision_Recognizer
-

Scope LatestAFPScope

ID :
LatestAFPScope
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
LatestAFPScope_Recognizer
-

Scope LatestATPScope

ID :
LatestATPScope
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
LatestATPScope_Recognizer
-

Scope PreviousRevision

ID :
PreviousRevision
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
PreviousRevision_Recognizer
-

Scope PreviousAFPScope

ID :
PreviousAFPScope
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
PreviousAFPScope_Recognizer
-

Scope PreviousATPScope

ID :
PreviousATPScope
Class :
kdm:code::AbstractCodeElement
Recognizer (if any) :
PreviousATPScope_Recognizer
-

Scope AddedTransactionalAFP

ID :
AddedTransactionalAFP
Class :
AFPTransactionHead
Recognizer (if any) :
AddedTransactionalAFP_Recognizer
-

Scope DeletedTransactionalAFP

ID :
DeletedTransactionalAFP
Class :
AFPTransactionHead
Recognizer (if any) :
DeletedTransactionalAFP_Recognizer
-

Scope UpdatedTransactionalAFP

ID :
UpdatedTransactionalAFP
Class :
AFPTransactionHead
Recognizer (if any) :
UpdatedTransactionalAFP_Recognizer
-

Scope DataAFPWithChangedType

ID :
DataAFPWithChangedType
Class :
DataAFP
Recognizer (if any) :
AFPDataWithChangedType_Recognizer
-

Scope SplitDataAFP

ID :
SplitDataAFP
Class :
DataAFP
Recognizer (if any) :
SplitDataAFP_Recognizer
-

Scope MergedDataAFP

ID :
MergedDataAFP
Class :
DataAFP
Recognizer (if any) :
MergedDataAFP_Recognizer
-

Scope AddedDataAFP

ID :
AddedDataAFP
Class :
DataAFP
Recognizer (if any) :
AddedDataAFP_Recognizer
-

Scope DeletedDataAFP

ID :
DeletedDataAFP
Class :
DataAFP
Recognizer (if any) :
DeletedDataAFP_Recognizer
-

Scope UpdatedDataAFP

ID :
UpdatedDataAFP
Class :
DataAFP
Recognizer (if any) :
UpdatedDataAFP_Recognizer
-

Scope EvolvedAFPArtifact

ID :
EvolvedAFPArtifact
Class :
AFPScope
Recognizer (if any) :
EvolvedAFPArtifact_Recognizer
-

Scope EvolvedATPArtifact

ID :
EvolvedATPArtifact
Class :
ATPScope
Recognizer (if any) :
EvolvedATPArtifact_Recognizer
-

Scope DETInUpdatedDataAFP_Previous

ID :
DETInUpdatedDataAFP_Previous
Class :
UpdatedDataAFP
Recognizer (if any) :
DETInUpdatedDataAFP_Previous_Recognizer
-

Scope DETInUpdatedDataAFP_Deleted

ID :
DETInUpdatedDataAFP_Deleted
Class :
UpdatedDataAFP
Recognizer (if any) :
DETInUpdatedDataAFP_Deleted_Recognizer
-

Scope DETInUpdatedDataAFP_Updated

ID :
DETInUpdatedDataAFP_Updated
Class :
UpdatedDataAFP
Recognizer (if any) :
DETInUpdatedDataAFP_Updated_Recognizer
-

Scope DETInUpdatedDataAFP_Added

ID :
DETInUpdatedDataAFP_Added
Class :
UpdatedDataAFP
Recognizer (if any) :
DETInUpdatedDataAFP_Added_Recognizer
-

Scope SharedArtifactInUpdatedTransactionalAFP_Deleted

ID :
SharedArtifactInUpdatedTransactionalAFP_Deleted
Class :
UpdatedTransactionalAFP
Recognizer (if any) :
SharedArtifactInUpdatedTransactionalAFP_Deleted_Recognizer
-

Scope SharedArtifactInUpdatedTransactionalAFP_Updated

ID :
SharedArtifactInUpdatedTransactionalAFP_Updated
Class :
UpdatedTransactionalAFP
Recognizer (if any) :
SharedArtifactInUpdatedTransactionalAFP_Updated_Recognizer
-

Scope SharedArtifactInUpdatedTransactionalAFP_Added

ID :
SharedArtifactInUpdatedTransactionalAFP_Added
Class :
UpdatedTransactionalAFP
Recognizer (if any) :
SharedArtifactInUpdatedTransactionalAFP_Added_Recognizer
-

Scope ArtifactInUpdatedTransactionalAFP_Deleted

ID :
ArtifactInUpdatedTransactionalAFP_Deleted
Class :
UpdatedTransactionalAFP
Recognizer (if any) :
ArtifactInUpdatedTransactionalAFP_Deleted_Recognizer
-

Scope ArtifactInUpdatedTransactionalAFP_Updated

ID :
ArtifactInUpdatedTransactionalAFP_Updated
Class :
UpdatedTransactionalAFP
Recognizer (if any) :
ArtifactInUpdatedTransactionalAFP_Updated_Recognizer
-

Scope ArtifactInUpdatedTransactionalAFP_Added

ID :
ArtifactInUpdatedTransactionalAFP_Added
Class :
UpdatedTransactionalAFP
Recognizer (if any) :
ArtifactInUpdatedTransactionalAFP_Added_Recognizer
-

Scope UpdatedTransactionalAFP_Latest

ID :
UpdatedTransactionalAFP_Latest
Class :
UpdatedTransactionalAFP
Recognizer (if any) :
UpdatedTransactionalAFP_Latest_Recognizer
-

Scope UpdatedTransactionalAFP_Previous

ID :
UpdatedTransactionalAFP_Previous
Class :
UpdatedTransactionalAFP
Recognizer (if any) :
UpdatedTransactionalAFP_Previous_Recognizer
-

Scope SQLQuery

ID :
SQLQuery
Class :
kdm:data::DataActions
Recognizer (if any) :
SQLQuery_Recognizer
-

7.19 List of parameter declarations

{ Real changedTypeCF = 0.40 }
{ Real mergedCF = 0.40 }
{ Real splitCF = 0.40 }
{ Real deletedCF = 0.40 }
{ Real addedCF = 1 }
{ Real wVeryHighEC = 1.2 }
{ Real wHighEC = 0.7 }
{ Real wModerateEC = 0.2 }
{ Real wLowEC = 0.1 }
{ Integer maxModerateSQLComplexity = 10 }
{ Integer maxHighSQLComplexity = 40 }
{ Integer maxVeryHighSQLComplexity = 70 }
{ Integer maxMediumLOC = 10 }
{ Integer maxLargeLOC = 50 }

{ Integer maxVeryLargeLOC = 200 }
{ Integer minModerateLackOfComment = 15 }
{ Integer minHighLackOfComment = 7 }
{ Integer minVeryHighLackOfComment = 3 }
{ Integer maxModerateFI = 4 }
{ Integer maxHighFI = 10 }
{ Integer maxVeryHighFI = 30 }
{ Integer maxModerateCC = 5 }
{ Integer maxHighCC = 15 }
{ Integer maxVeryHighCC = 30 }
{ Integer maxUpdateSQLStatement = 0 }
{ Integer wUpdateSQLStatement = 10 }
{ Integer maxGroupBySQLStatement = 0 }
{ Integer wGroupBySQLStatement = 0 }
{ Integer maxSQLSubqueries = 0 }
{ Integer wSQLSubqueries = 10 }
{ Integer maxSQLTableColumns = 9 }
{ Integer wSQLTableColumns = 10 }
{ Integer maxSQLTables = 30 }
{ Integer wSQLTables = 10 }
{ Integer maxSQLTablesPerQuery = 4 }
{ Integer wSQLTablesPerQuery = 50 }

7.20 List of external references

Automated Function Points 1.0

afp:afp::EI
afp:afp::Element
afp:afp::EO
afp:afp::isEI
afp:afp::isEIF
afp:afp::isEO
afp:afp::isILF
afp:afp::LF
afp:afp::wEI
afp:afp::wEIF
afp:afp::wEO
afp:afp::wILF

Knowledge Discovery Meta-Model 1.3

kdm:action::AbstractActionRelationship
kdm:action::CallableRelations
kdm:action::DataRelations

kdm:code::AbstractCodeElement
kdm:code::ControlElement
kdm:code::DataElement
kdm:code::Extends
kdm:code::Implements
kdm:Core::Element
kdm:data::DataActions
kdm:data::WritesColumnSet
kdm:data::ReadsColumnSet
kdm:kdm::Segment
kdm:ui::AbstractUIElement

Structured Metric Meta-Model 1.0

smm:SMMSampleLibrary::CodeEltTotalLOC
smm:SMMSampleLibrary::CodeEltTotalLOComment
smm:SMMSampleLibrary::McCabeCyclomaticComplexity

7.21 Output Generation

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

- What is the functional size of the software enhancement? What is the functional size of feature enhancements?
- What is the implementation size of the software enhancement? What is the implementation size of feature enhancements?
- Where are the software enhancement taking place?
- What are the assumptions used in the process?

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:

- Automated Enhancement Points (AEP) value
- Automated Enhancement Function Points (AEFP) and Automated Enhancement Technical Points (AETP) values
- Implementation Points values for Automated Enhancement Function Points and Automated Enhancement Technical Points scopes (IP(AEFP) and IP(AETP))
- Location of each Data Function and Transactional Function underlying elements in the source code (Data AFP Implementation Scopes and Transactional AFP Implementation Scopes),
    - with their evolution status (added, deleted, updated, unchanged),
    - with their sharing status,
    - and with their Artifact nature (Transactional AFP Implementation Artifacts)

- Effort Complexity of each Artifact, with the details of the underlying category assignments (Cyclomatic Complexity level, Size level, Comment level, SQL Complexity level and value, Coupling level)
- List of evolved Data AFP (added, deleted, modified, merged, split, with changed type) and Transactional AFP (added, deleted, modified), with their Complexity Factor
    - In case of variable value – that is, for each modified Data and Transactional AFP – the Complexity Factor input – that is, for each modified Data AFP, the percentage of changed DET and for each modified Transactional AFP, the percentage of Effort Complexity variation for all Artifacts in the Implementation Scope, the percentage of Effort Complexity in evolved Artifacts, and the percentage of Effort Complexity in evolved shared Artifacts –
- A complete list of inputs used by to generate the outputs.

8. Usage Scenarios (Informative)

The usage scenarios following the applicable use cases are the following

- Gain visibility into the amount of enhanced or changed functional features between selected revisions or releases of an application
- Gain visibility into the implementation complexity of the enhanced or changed functional features between selected revisions or releases of an application
- Gain visibility into the implementation complexity of the enhanced or changed technical foundation software between selected revisions or releases of an application
- Get the functional equivalent amount of enhanced or changed technical foundation software between selected revisions or releases of an application
- Get the productivity of maintenance and enhancement work between selected revisions or releases of an application
- Get the functional productivity between selected revisions of an application

8.1 Delivered Amount of Software Features Enhancement

To get a measurement of the amount of enhanced or changed functional features (those traditionally measured with Automated Function Points between two software revisions, use the Automated Enhancement Function Points value which:

- focuses only on the enhancements that impact directly the functional features traditionally measured with by Automated Function Points, overlooking the enhancements that impact the software technical foundation, and
- is expressed in an Enhancement Function Point units, which are aligned with the Automated Function Points unit in terms of concept and magnitude.

This should aid consumption of the results by business-oriented audiences, whose focus is on functional feature evolutions only, and not of the required amount of work.

With Automated Enhancement Function Points, managers can monitor and benchmark how maintenance and enhancement projects impact the functional features of an application. This measure enables them to define a measure for functional productivity for maintenance and enhancement work as described in clause 8.2.

8.2 Overall Functional Enhancement Productivity

To get a measurement for the productivity of implementing functional enhancements, simply divide the Automated Enhancement Function Points value by the Effort expended on the work. Note that in this scenario:

- the Effort value has to consider the same scope of work as is measured by the Automated Enhancement Function Points Scope,
- this productivity measure only relates to functional maintenance and enhancement work, and does not account for maintenance and enhancements on the technical foundation software.

8.3 Implementation Complexity of Enhancing Software Features

To get a measurement of the implementation complexity for maintenance and enhancement work on functional features between two software releases, use of Implementation Points calculated on the Automated Enhancement Function Points value which:

- focuses only on the enhancements that impact directly the functional features traditionally measured with Automated Function Points, overlooking changes that impact the software technical foundation,
- but takes into account the complexity of implementing the changes to account for the amount of work required to enhance the functional features.

With Automated Enhancement Function Points and their Implementation Points value, managers can monitor and benchmark the impact of maintenance and enhancement projects on the functional features, taking into account the required Effort. This enables an analysis of Return on Investment for functional features, but these analyses will not reflect the value of improving the technical foundation of the application.

8.4 Implementation Complexity of Enhancing Technical Foundation Software

To measure the implementation complexity of maintenance and enhancement work on the technical foundation software between two software releases, use Implementation Points calculated on the Automated Enhancement Technical Points value which:

- focuses only on the enhancements that impact directly the technical foundation software, overlooking the enhancements that impact the functional features,
- takes into account the complexity of implementing these changes to account for the amount of work required to enhance the technical foundation software.

8.5 Functional vs. Technical Investment Ratio

This ratio comparing the amount of change in functional software to that in technical foundation software helps educate the business on where, how, and why development investment dollars/Euros/etc. are spent. With Automated Enhancement Function Points and the Implementation Points derived from both the Automated Enhancement Function Points value and the Automated Enhancement Technical Points value, managers can monitor and benchmark not only how maintenance and enhancement projects impact the software features, but also how much change this requires in the technical foundation software as well. However, this ratio does not provide a direct measure of the amount of technical work that can be easily understood by the business-oriented audiences.

The formula for measuring the share of maintenance and enhancement activities that deal with software features is based in Implementation Points and is computed as IP_{AEFP} / (IP_{AETP} + IP_{AEFP}). Similarly, the formula for measuring the share of maintenance and enhancement activities that deal with technical foundation software is computed as IP_{AETP} / (IP_{AETP} + IP_{AEFP}). This ratio is an objective measure that helps explain how much work will be required on the technical foundation software (as deemed

necessary by the software architects and development teams) to support the functional evolutions requested by the lines of business.

8.6 Functional Equivalent Amount of Technical Foundation Software Enhancement

To get a measurement of the technical foundation software enhancements between two software releases that is statistically equivalent Automated Function Points, use of Automated Enhancement Technical Points value adjusted by the Equivalence Ration which:

- focuses only on the enhancements that impact exclusively the technical foundation software, overlooking the enhancements that impact the functional features,
- Is expressed in a AFP_{eq} units, which is statistically equivalent to Automated Function Point units in terms of magnitude,

This measure helps ease the consumption of the results by business-oriented audiences, who focus on visible feature evolutions only (and not of the technical work on the technical foundation software that does not translate into enhanced software features).

8.7 Overall Enhancement Implementation Productivity

Automated Enhancement Function Points, Automated Enhancement Technical Points, and the Implementation Point measures based on each or them, managers can monitor and benchmark all aspects of the maintenance and enhancement work performed on an application, using raw or AFP-equivalent units to adapt to the different audiences. These measures also support the calculation of Automated Enhancement Points which enables a productivity indicator for maintenance and enhancement work defined as Automated Enhancement Points / Effort. In this scenario, it is key to note that:

- The Effort value has to match the same scopes as Automated Enhancement Function Points Scope, Automated Enhancement Technical Points Scope,
- This is overall maintenance and enhancement productivity, which accounts for work on both the functional features of an application and the technical foundation software that supports them. This provides a better accounting of the work produced by the total effort spent, and is therefore a superior productivity measure for maintenance and development work.

9. References

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

IFPUG (2014). *Software Non-functional Assessment Practices Manual 2.1*, Princeton Junction, NJ: International Function Point Users Group.

ISO (2009). *ISO/IEC 20926:2009 — Software measurement — IFPUG Functional Size Measurement Method 2009.* Geneva: International Organization for Standards.

McCabe, T. (1976). A measure of software complexity. *IEEE Transactions on Software Engineering*.

NESMA (2009). *Function Point Analysis for Software Enhancement, Version 2.2.1*. Amsterdam: Netherlands Software Measurement Users Association,

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

Appendix A: Artifact Data Types

SQL Languages

- Function
- Procedure
- Package function
- Package procedure
- Trigger
- View

C/C++ Language

- C/C++ Function
- C/C++ Method
- C/C++ Constructor
- C/C++ Destructor

Notes:

- C/C++ Macros are not counted as Artifacts as they are most of the time a single line of code shortcut.
- Only C++ objects with a code definition are considered to be artifacts
- C++ Methods/Function declared in a .h but not implemented in a C++ file are not considered to be artifacts

Visual Basic Language

- VB Event
- VB Function
- VB Property Get
- VB Property Let
- VB Property Set
- VB Sub

Java Language

- Java Constructor
- Java Method
- Java Initializer

Notes:

- The automatically generated Java Methods like '_jspService' are not considered as Artifacts.

- The Java Classes that belong to standard libraries and custom libraries (i.e. their source code is not available) are not considered as Artifacts.

Mainframe

- Cobol Program
- JCL Job
- JCL Procedure
- IMS Segment
- IMS DB PCB

MS.NET Languages

- Method
- Property Set
- Property Get
- AddOn
- RemoveOn
- Fire
- Constructor
- Destructor
- Event
- eFunction
- eSub
- ePropertyGet
- ePropertySet
- ePropertyLet
- eEvent
- eFile

Web Languages (JSP/ASP/JS)

- Method
- eFunction
- eSub
- eMethod
- ePropertyGet
- ePropertySet
- ePropertyLet
- eEvent
- Java Method
- eFile

Notes:

- eFiles are included in the count only when they contain executable source code (file extensions like *.jsp, *.asp, *.js), excluding from the list HTML files and other images, configuration files …

SAP ABAP

- ABAP Form
- ABAP Function
- ABAP Event Block
- ABAP Module
- ABAP Method
- ABAP Constructor
- ABAP Event Method
- ABAP File Level Code of custom programs, user-exits and includes
- ABAP Event
- WebDynpro Supply Function
- WebDynpro Event Handler
- WebDynpro Method

Appendix B: 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. Each software quality characteristic is shown with the sub-characteristics that compose it.