INTERNATIONAL STANDARD

ISO/IEC 19515

First edition
2019-05

Information technology — Object Management Group Automated Function Points (AFP), 1.0

Reference number
ISO/IEC 19515:2019(E)

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

Published in Switzerland

Contents

Page

Foreword .... **iv**

Introduction .... **v**

1 Scope **1**
- 1.1 Purpose 1
- 1.2 Applicability 1
- 1.3 Limitations 1

2 Conformance and Compliance **1**
- 2.1 Conformance 1
- 2.2 Compliance 2
- 2.3 Consistency with IFPUG CPM 2

3 References **3**
- 3.1 Normative 3
- 3.2 Non-normative 3

4 Terms and Definitions **3**

5 Symbols (and abbreviated terms) **5**

6 Additional Information **5**
- 6.1 Overview of Function Points 5
- 6.2 Function Point Usage Scenarios 6
- 6.3 Inputs to Automated Function Point Counting 7
- 6.4 Outline of the Function Point Counting Process 7
- 6.5 The Application Model 8
  - 6.5.1 The Application Model Elements 8
  - 6.5.2 Detection of Data Functions 9
  - 6.5.3 Detection of Transactional Functions 14
  - 6.5.4 Detection of Internal Versus External Logical Files 15

7 Determine Functional Size (Normative) **17**
- 7.1 Entering Application Model Elements into Functional Sizing 17
  - 7.1.1 Representation of the Application Model in KDM 17
  - 7.1.2 Translating KDM Application Model Elements into SMM Inputs 18
- 7.2 Determine Data Function Size 18
- 7.3 Determine Transactional Function Size 19
- 7.4 Determine Function Point Size 22
- 7.5 Output Generation 22
- 7.6 Structured Metrics Meta-Model (SMM) Representation 23
  - 7.6.1 Computing Automated Function Point Size 23
  - 7.6.2 Computing External Output Size 24
  - 7.6.3 Computing External Input Size 25
  - 7.6.4 Computing Internal Logical File Size and External Interface File Size 26

8 References **27**

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by the Object Management Group (OMG) (as the OMG specification for Automated Function Points (AFP), v1.0) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

This document is related to:

— ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2:1995, *Information Technology — Open Distributed Processing — Reference Model: Foundations*

— ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3:1995, *Information Technology — Open Distributed Processing — Reference Model: Architecture*

— ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1997, *Information Technology — Open Distributed Processing — Interface Definition Language*

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The rapid growth of distributed processing has led to a need for a coordinating framework for this standardization and ITU-T Recommendations X.901-904 | ISO/IEC 10746, the Reference Model of Open Distributed Processing (RM-ODP) provides such a framework. It defines an architecture within which support of distribution, interoperability and portability can be integrated.

RM-ODP Part 2 (ISO/IEC 10746-2) defines the foundational concepts and modeling framework for describing distributed systems. The scopes and objectives of the RM-ODP Part 2 and the UML, while related, are not the same and, in a number of cases, the RM-ODP Part 2 and the UML specification use the same term for concepts which are related but not identical (e.g., interface). Nevertheless, a specification using the Part 2 modeling concepts can be expressed using UML with appropriate extensions (using stereotypes, tags, and constraints).

RM-ODP Part 3 (ISO/IEC 10746-3) specifies a generic architecture of open distributed systems, expressed using the foundational concepts and framework defined in Part 2. Given the relation between UML as a modeling language and Part 3 of the RM-ODP standard, it is easy to show that UML is suitable as a notation for the individual viewpoint specifications defined by the RM-ODP.

This International Standard defines a method for automating the counting of Function Points that is generally consistent with the Function Point Counting Practices Manual, Release 4.3.1 (IFPUG CPM) produced by the International Function Point Users Group (IFPUG). Guidelines in this specification may differ from those in the IFPUG CPM at points where subjective judgments have to be replaced by the rules needed for automation. The IFPUG CPM was selected as the anchor for this specification because it is the most widely used functional measurement specification with a large supporting infrastructure maintained by a professional organization.

Information technology — Object Management Group Automated Function Points (AFP), 1.0

1 Scope

1.1 Purpose

This International Standard defines a method for automating the counting of Function Points that is generally consistent with the Function Point Counting Practices Manual, Release 4.3.1 (IFPUG CPM) produced by the International Function Point Users Group (IFPUG). Guidelines in this International Standard may differ from those in the IFPUG CPM at points where subjective judgments have to be replaced by the rules needed for automation. The IFPUG CPM was selected as the anchor for this International Standard because it is the most widely used functional measurement specification with a large supporting infrastructure maintained by a professional organization.

1.2 Applicability

This International Standard is applicable to the functional sizing of transaction-oriented software applications, and in particular those with data persistency. To be consistent with the IFPUG CPM, the International Standard provides details on the support of applications using relational databases. However, the International Standard can be used and extended for any type of transactional application with data persistency.

1.3 Limitations

This International Standard does not address the sizing of enhancements to an application or maintained functionality (often called Enhancement Function Points). Extensions of the automated counting methods described in this International Standard such as Automated Enhancement Function Points will be addressed in future addendums to this International Standard. This International Standard does not address sizing for the non-functional components of a software application. Non-functional components (as defined by IFPUG) include:

— Structural Quality Constraints Reliability, Security, Performance Efficiency, Maintainability, etc.

— Organizational Constraints locations for operations, target hardware, compliance to standards, etc.

— Environmental Constraints interoperability, security, privacy, safety, etc.

— Implementation Constraints development language, delivery schedule, etc.

2 Conformance and Compliance

2.1 Conformance

This International Standard is derived from IFPUG's Function Point Counting Practices Manual, Release 4.3.1 (IFPUG CPM). However, explicit counting rules were specified in this International Standard in order to provide for rigorous automation that may not be in strict conformance with guidance in IFPUG's manual. Therefore, there is no claim of strict conformance with the IFPUG CPM standard. Additionally, this International Standard has made every attempt to conform to the extent possible with international standards for functional measurement, in particular ISO/IEC 25010, ISO/IEC 20926:2009, and NEN-ISO/IEC 24570. This International Standard conforms to OMG's Knowledge Discovery Meta-model (KDM) and Structured Metrics Meta-model (SMM) in its specification and representation of the Automated Function Point counting and scoring process.

Conformance with this International Standard by Automated Function Point counting tools is determined by analyzing the KDM elements that constitute the Application Model and produce the final Automated Function Point count using the counting process specified in the SMM model presented in this International Standard. Output from this automated process should conform to the list of output artifacts listed in Clause 7.

2.2 Compliance

Implementations of this International Standard should be able to demonstrate the following attributes in order to claim compliance:

— Automated — Although the initial inputs such as the source code, definition of application boundary, and some naming conventions are provided manually to initiate Automated Function Point counting, the analysis of the source code and the actual counting must be fully automated.

— Consistent — Two independent and separate functional sizings performed on the same application source code using the same boundaries and other required manual inputs by different Automated Function Point tools that conform to this International Standard must produce the same results in terms of Automated Function Point size (i.e., the same number of Automated Function Points).

— Verifiable — Implementations that comply with this International Standard must clearly list each and every input the implementation requires and list each and every output that the implementation generates so that they can be audited by a third party. Implementations should provide a list of assumptions/heuristics (to the extent that this does not disclose proprietary information) used to transform the inputs to the outputs so that the calculations can be independently verified by third parties.

2.3 Consistency with IFPUG CPM

This International Standard for Automated Function Points follows the steps for the counting process in the IFPUG CPM to the extent possible for an automated system. An automated system relies on receiving the correct and complete list of inputs to fulfill this step. Consequently the initial step in the Automated Function Point counting process is the manual gathering of inputs and configuring them properly for analysis by the Automated Function Point counting technology. The remainder of the analysis and counting process is automated.

This International Standard prioritizes repeatability and consistency over consistency with the IFPUG CPM counting guidelines. In some counting situations, IFPUG guidelines are vague, leaving the interpretation to the judgment of the counter. Consequently, IFPUG certified Function Point counters often differ by as much as 10 % or more in the counts they produce. In order to remove subjectivity, this International Standard makes explicit decisions about counting techniques in situations where the IFPUG guidelines were vague. Consequently we have introduced some variation from the IFPUG guidelines in order to achieve the precision required for automation and repeatability.

Automated Function Point counts may differ from the manual counts produced by IFPUG Certified Function Point counters. In fact, manual counts produced by different IFPUG Certified Function Point counters on the same code base will typically differ because of different interpretations of the designer's or developer's intent in creating various functional elements of an application. In the manual IFPUG counting process, documentation, and expert knowledge are typically used in sizing an application. These types of inputs are not available to an automated system. Hence, any automated approach will be unable to capture information about the designer's or developer's intent that isn't explicitly encoded in the application's source code. Since this International Standard relies exclusively on the application's source code as defined by its boundaries, it cannot capture any of the designer's intentions that do not leave an 'imprint' on the source code. Some of these intentions might make a difference in manual counting. For this additional reason, the Automated Function Point counts produced by technology that is compliant with this International Standard may differ from manual counts produced by IFPUG Certified Function Point counters. The advantage of Automated Function Points is that a tool will produce repeatable, consistent counts, attributes that are not characteristic of manual counting.

3 References

3.1 Normative

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

— Structured Metrics Meta-model, version 1.0 (SMM), formal/2012-01-05

— Knowledge Discovery Meta-model, version 1.3 (KDM), formal/2011-08-04

— Unified Modeling Language, version 2.4.1 (UML), formal/2011-08-05

— MOF/XMI Mapping, version 2.4.1 (XMI), formal/2011-08-09

3.2 Non-normative

— Function Point Counting Practices Manual, Release 4.3.1. ISBN 978-0-9753783-4-2

— Function Point Analysis. ISBN 0-201-69944-3

4 Terms and Definitions

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

Application Model
abstract source code representation of an application that results from analysis of the source code. It contains the minimum information required to measure Automated Function Points, that is, the static elements of an application defined by associated KDM elements that are used in the Automated Function Point counting process.

Boundary
the boundary is a conceptual interface between the software and the users. (IFPUG CPM)

Complete transaction
in the context of this International Standard, a transaction is considered as complete whenever the static code analyzer can find one or several code paths starting from the user interface and continuing down to the data entities.

Data Element Type (DET)
a data element type is a unique user recognizable, non-repeated attribute that is part of an ILF, EIF, EI, or EO. (ISO/IEC 20926:2009)

Data Function
functionality provided to the user to meet internal or external data storage requirements. A data function is either an ILF or EIF. (ISO/IEC 20926:2009)

Database Table
SQL data table or KDM's data:RelationalTable.

External Inquiry (EQ)
a form of data that is leaving the system. However, since automated counting tools cannot distinguish between External Inquiries and External Outputs, all External Inquiries will be included in and counted as External Outputs. (ISO/IEC 20926:2009)

External Input (EI)
elementary process that processes data or control information sent from outside the boundary (ISO). (ISO/IEC 20926:2009)

External Interface File (EIF)
a user recognizable group of logically related data or control information, which is referenced by the application being measured, but which is maintained within the boundary of another application. (ISO/IEC 20926:2009)

External Output (EO)
an elementary process that sends data or control information outside the boundary and includes additional processing logic beyond that of an External Inquiry. (ISO/IEC 20926:2009)

File Type Referenced (FTR)
data function (IFL or EIF) read and/or maintained by a transactional function. (ISO/IEC 20926:2009)

Internal Logical File (ILF)
user recognizable group of logically related data or control information maintained within the boundary of the application being measured. (ISO/IEC 20926:2009)

Library
a set of software components that are grouped together in the same physical container and that are accessed via a dedicated API.

Logical File
either an Internal Logical File (ILF) or an External Interface File (EIF). (ISO/IEC 20926:2009)

Method
a method is a group of instructions that is given a name and can be called up at any point in a program simply by quoting that name. In object oriented languages like Java and C++, methods are grouped in classes. A method is referenced as code:MethodUnit in the KDM.

Physical File
physical files hold the actual data of a database file and are not required to have keyed fields. Where the word 'file' is used alone without the modifier 'logical,' it refers to a physical file. Within the KDM, such a physical file is described as a kind of data:DataContainer and contains a set of data:RecordFile.

Record Element Type (RET)
user recognizable sub-group of data element types within a data function. (ISO/IEC 20926:2009)

Service End Point
well known address that is used by an application to exchange data and events with other applications. Typical examples include Remote Procedure Call interfaces and Message Queues.

Source Code Entities
elements of the source that can be detected during static analysis in order that they be used in the Function Point counting process.

Structured Query Language (SQL)
a language used to query databases. (ISO/IEC 9075-1:2008)

Static Dependency
a directional relation that exists between a caller method and a called method.

Transaction End Point
user Interface End Point or a Service End Point. It identifies potential Transactional Functions.

Transactional Function
elementary process that provides functionality to the user to process data. A transactional function is an External Input, External Output, or External Inquiry. (ISO/IEC 20926:2009)

User Interface End Point
there are two kinds of user interface end points: user interface inputs and user interface outputs.

User Interface Input
command that can be activated by humans using a mouse, keyboard, or equivalent interactions. (ISO/IEC 20926:2009)

User Interface Output
set of visual elements that are composed by an application in order to present information or events to the user (e.g., a form, report, or tab inside a screen). An elementary process that sends data or control information outside the boundary. (ISO/IEC 20926:2009)

User Recognizable
refers to requirements for processes and/or data that are agreed upon, and understood by, both the user(s) and software developer(s). (IFPUG CPM)

5 Symbols (and abbreviated terms)

APF Automated Function Point

CISQ Consortium for IT Software Quality

CPM Counting Practices Manual

DET Data Element Type

DFC Data Function Complexity

EQ External Inquiry

EI External Input

EIC External Input Complexity

EIF External Interface File

EO External Output

EOC External Output Complexity

FTR File Type Referenced

IFPUG International Function Point Users Group

ILF Internal Logical File

KDM Knowledge Discovery Meta-model

RET Record Element Type

SMM Structured Metrics Meta-model

SQL Structured Query Language

6 Additional Information

6.1 Overview of Function Points

The use of Function Points as a measure of the functional size of software was initially introduced in the mid-1970s and today is used by organizations worldwide. Allan Albrecht of IBM was the first to publicly release a method for functionally sizing software called Function Point Analysis (Albrecht, 1979, 1981).

Since its formation in 1986 the International Function Point Users Group (IFPUG) has continuously maintained and enhanced the original Albrecht method for functionally sizing software (IFPUG CPM).

Function Points are a normalized metric that can be used consistently with an acceptable degree of accuracy. The value of Function Point analysis centers on its ability to measure the size of any software deliverable in logical, user-oriented terms. Function Point counts are technology agnostic in that they do not depend on the type of technology or development language. Function Points simply measure the functionality that an application delivers to an end user.

Function Point analysis evaluates a software deliverable and measures its size based on well-defined functional characteristics of a software system. Function Point analysis accounts for four constituents of an application:

1) External Inputs — Input data that is entering a system (logical transaction inputs, system feeds).

2) External Outputs and External Inquires — data that is leaving the system (on-line displays, reports, feeds to other systems).

3) Internal Logical Files — data that is processed and stored within the system (logical groups of user defined data).

4) External Interface Files — data that is maintained outside the system but is necessary to satisfy a particular process requirement (interfaces to other systems).

Organizations can apply function points to measure the size of a software product. Along with selected other measures, Function Points can be used in the following activities:

— Quality and productivity analysis.

— Estimating the costs and resources required for software development, enhancement, and maintenance.

— Normalizing data used in software comparisons.

— Determining the size of a purchased application package (Commercial Off The Shelf or customized system) by sizing all the functionality included in the package.

— Enabling users to determine the Return on Investment of an application by sizing the functionality that specifically matches the requirements of their organization.

6.2 Function Point Usage Scenarios

There are three frequent scenarios of use for Function Points; estimating software size in order to estimate effort and cost, normalization of other measures, and benchmarking for decision-making.

1) Function Points have been defined around components that can be identified in a well-written specification. Consequently Function Points can be counted from the specification well before other size measures such as lines of code become available. Because of this advantage many commercial cost estimating formulas use Function Points in their calculations as the preferred size measure from which effort and cost figures can be estimated. When the actual number of Function Points becomes available from the delivered software, parameters in these formulas can be adjusted to increase their accuracy.

2) Function Points are frequently used to normalize other measures. For instance, the total defects detected in a software system can be normalized by Function Points to provide a measure of the defect density per Function Point. This allows better comparison among systems that differ in size, as well as programming language.

3) Because they are counted from constructs that are independent of the programming language, Function Points are a preferred method for comparing software systems when benchmarking. In particular they are frequently used to compare the past performance and productivity

