INTERNATIONAL STANDARD

ISO/IEC 20926

Second edition
2009-12-01

Software and systems engineering — Software measurement — IFPUG functional size measurement method 2009

Ingénierie du logiciel et des systèmes — Mesurage du logiciel — Méthode IFPUG 2009 de mesurage de la taille fonctionnelle

Reference number
ISO/IEC 20926:2009(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 20926 was prepared by the International Function Point Users Group (IFPUG) and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

This second edition cancels and replaces the first edition (ISO/IEC 20926:2003), which has been technically revised.

Introduction

The use of function points, as a measure of the functional size of software, has grown since the mid 1970s from a few interested organizations to an impressive list of organizations worldwide. Allan Albrecht was the first to publicly release a method for functionally sizing software called function point analysis. With the growth in the use of function points, there has been wider application and use of the measure. Since its formation in 1986 the International Function Point Users Group (IFPUG) has continuously enhanced the original Albrecht method for functionally sizing software. This International Standard is the latest release in the continually improving IFPUG method that promotes the consistent interpretation of functional size measurement in conformance with ISO/IEC 14143-1:2007. The IFPUG functional size measurement method is known as function point analysis and its units of functional size are called Function Points.

Organizations can apply this International Standard to measure the size of a software product to:

— support quality and productivity analysis;

— estimate cost and resources required for software development, enhancement and maintenance;

— provide a normalization factor for software comparison;

— determine the size of a purchased application package by functionally sizing all the functions included in the package;

— assist users in determining the benefit of an application package to their organization by functionally sizing functions that specifically match their requirements.

Function point analysis measures software by quantifying the tasks and services (i.e., functionality) that the software provides to the user based primarily on logical design. The objectives of function point analysis are to measure:

— functionality implemented in software, that the user requests and receives;

— functionality impacted by software development, enhancement and maintenance independently of technology used for implementation.

The process of function point analysis is:

— simple enough to minimize the overhead of the measurement process;

— a consistent measure among various projects and organizations.

In order to effectively apply this International Standard, persons can be formally trained in the method using IFPUG certified course materials.

This International Standard is one component in the IFPUG publications. It is recommended that it be read in conjunction with the other IFPUG publications. These provide guidance to application of the rules specified within this International Standard and background information to aid in understanding the use and applicability of the resulting functional size. Supporting IFPUG publications include the following:

— the current version of the IFPUG Counting Practices Manual, which incorporates this International Standard supplemented with counting practices and examples that support its implementation;

— “Framework for Functional Sizing”, 2005, which discusses the contribution of both functional size and non-functional size to the overall software product size; the IFPUG FSM method is a method for measuring the functional size;

— IFPUG website at www.ifpug.org.

Software and systems engineering — Software measurement — IFPUG functional size measurement method 2009

1 Scope

1.1 Purpose

This International Standard specifies the set of definitions, rules and steps for applying the IFPUG functional size measurement (FSM) method.

1.2 Conformity

This International Standard is conformant with all mandatory provisions of ISO/IEC 14143-1:2007.

1.3 Applicability

This International Standard can be applied to all functional domains.

NOTE IFPUG continues to publish white papers providing guidelines for use in evolving environments and domains.

This International Standard is fully convertible to prior editions of IFPUG sizing methods.

IFPUG function point analysts have identified different delivery rates (hours to deliver a function point) related to building applications in different functional domains calibrated for varying project sizes and software complexities.

1.4 Audience

This International Standard can be applied by anyone requiring a measurement of functional size. Persons experienced with the method will find this International Standard to be a useful reference.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14143-1:2007, *Information technology — Software measurement — Functional size measurement — Part 1: Definition of concepts*

3 Terms and definitions

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

3.1
adaptive maintenance
modification of a software product, performed after delivery, to keep a software product usable in a changed or changing environment

NOTE Adaptive maintenance provides enhancements necessary to accommodate changes in the environment in which a software product must operate. These changes are those that must be made to keep pace with the changing environment. For example, the operating system might be upgraded and some changes may be made to accommodate the new operating system.

[ISO/IEC 14764:2006, 3.1]

3.2
application
cohesive collection of automated procedures and data supporting a business objective, consisting of one or more components, modules, or subsystems

EXAMPLES Accounts payable, accounts receivable, payroll, procurement, shop production, assembly line control, air search radar, target tracking, weapons firing, flight line scheduling and passenger reservations.

3.3
application functional size
measure of the functionality that an application provides to the user, determined by the application function point count

3.4
application function point count
activity of applying this International Standard to measure the functional size of an application

3.5
arranging
activity of sequencing attributes in a transactional function

3.6
associative entity type
entity type that contains attributes which further describe a many-to-many relationship between two other entity types

3.7
attributive entity type
entity type that further describes one or more attributes of another entity type

3.8
base functional component
BFC
elementary unit of Functional User Requirements defined by and used by an FSM Method for measurement purposes

EXAMPLE A Functional User Requirement could be “Maintain Customers”, which might consist of the following BFCs: “Add a new customer”, “Report Customer Purchases”, and “Change Customer Details”. Another example might include a collection of logically related business data maintained by the software under study such as “Customer Details”. There are many other examples.

[ISO/IEC 14143-1:2007, 3.1]

3.9
boundary
conceptual interface between the software under study and its users

[ISO/IEC 14143-1:2007, 3.3]

NOTE ISO/IEC 20926:2003 used the term “application boundary”.

3.10
consistent state
point at which processing has been fully executed, the Functional User Requirement has been satisfied and there is nothing more to be done

EXAMPLE 1 The Functional User Requirement is to print a check and mark the appropriate account as paid. If only part of the Functional User Requirement is completed (e.g., only printing the check or only marking it as paid), the application is not in a consistent state. The printing of a check without marking the account as paid causes an inconsistency in the application as does marking it as paid without printing it.

EXAMPLE 2 The Functional User Requirement is to have a batch process that accepts an input file to update a data store, produce a production control report and return an error report back to the sending application. The process is not in a consistent state unless all of these parts are completed.

EXAMPLE 3 The Functional User Requirement is to transfer an employee to a new job and validate his/her security clearance level. To complete this, a real-time request is sent to the security application (which maintains government security clearances and not application security) and a response received before the transfer can be completed. All steps are needed to create a consistent state. The interaction with the security application is not an independent step or action. It does not happen in and of itself, and the transaction to transfer an employee is not in a consistent state without it.

3.11
control information
data that influences an elementary process by specifying what, when or how data is to be processed

3.12
conversion functionality
transactional or data functions provided to convert data and/or provide other user specified conversion requirements

NOTE Conversion functionality exists only during the development or enhancement of an application.

3.13
corrective maintenance
reactive modification of a software product performed after delivery to correct discovered problems

NOTE The modification repairs the software product to satisfy requirements.

[ISO/IEC 14764:2006, 3.2]

3.14
counting scope
set of Functional User Requirements to be included in the function point count

3.15
Data Element Type
DET
unique, user recognizable, non-repeated attribute

3.16
data function
functionality provided to the user to meet internal or external data storage requirements

NOTE A data function is either an Internal Logical File or an External Interface File.

3.17
derived data
data created as a result of processing that involves steps other than or in addition to direct retrieval and validation of information from data functions

3.18
development project
project to develop and deliver the first release of a software application

3.19
development project functional size
measure of the functionality provided to the users with the first release of the software, as measured by the development project function point count

NOTE The functional size of a development project can include the size of conversion functionality.

3.20
development project function point count
activity of applying this International Standard to measure the functional size of a development project

3.21
elementary process
smallest unit of activity that is meaningful to the user

3.22
enhancement project
project to develop and deliver adaptive maintenance

NOTE An enhancement project can also develop and deliver corrective and perfective maintenance, but these do not contribute to the enhancement project functional size.

3.23
enhancement project functional size
measure of the functionality added, changed or deleted at the completion of an enhancement project, as measured by the enhancement project function point count

NOTE The functional size of an enhancement project can include the size of conversion functionality.

3.24
enhancement project function point count
activity of applying this International Standard to measure the functional size of an enhancement project

3.25
entity dependent
⟨entity⟩ not meaningful or not significant to the business in and of itself without the presence of other entities, such that

— an occurrence of entity X must be linked to an occurrence of entity Y, and

— the deletion of an occurrence of entity Y results in the deletion of all related occurrences of entity X

3.26
entity independent
⟨entity⟩ meaningful or significant to the business in and of itself without the presence of other entities

3.27
External Input
EI
elementary process that processes data or control information sent from outside the boundary

NOTE An External Input is a type of base functional component.

3.28
External Inquiry
EQ
elementary process that sends data or control information outside the boundary

NOTE An External Inquiry is a type of base functional component.

3.29
External Interface File
EIF
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

NOTE An External Interface File is a type of base functional component.

3.30
External Output
EO
elementary process that sends data or control information outside the boundary and includes additional processing logic beyond that of an External Inquiry

NOTE An External Output is a type of base functional component.

3.31
File Type Referenced
FTR
data function read and/or maintained by a transactional function

3.32
functional complexity
specific complexity rating assigned to a function using the rules as defined within this International Standard

3.33
functional size
size of the software derived by quantifying the Functional User Requirements

[ISO/IEC 14143-1:2007, 3.6]

3.34
Functional User Requirements
sub-set of the user requirements specifying what the software shall do in terms of tasks and services

NOTE 1 Functional User Requirements include, but are not limited to, the following:

— data transfer (for example: Input customer data, Send control signal);
— data transformation (for example: Calculate bank interest, Derive average temperature);
— data storage (for example: Store customer order, Record ambient temperature over time);
— data retrieval (for example: List current employees, Retrieve aircraft position).

User requirements that are not Functional User Requirements include, but are not limited to, the following:

— quality constraints (for example: usability, reliability, efficiency and portability);
— organizational constraints (for example: locations for operation, target hardware and compliance to standards);
— environmental constraints (for example: interoperability, security, privacy and safety);
— implementation constraints (for example: development language, delivery schedule).

NOTE 2 Adapted from ISO/IEC 14143-1:2007, 3.8.

3.35
Function Point
FP
unit of measure for functional size as defined within this International Standard

3.36
Function Point Analysis
FPA
method for measuring functional size as defined within this International Standard

3.37
function point count
activity of applying the rules within this International Standard to measure the functional size of an application or project

NOTE There are three types of function point count: application, development project and enhancement project.

3.38
function type
type of base functional component identified in this International Standard

NOTE The five function types identified in this International Standard are: External Input, External Output, External Inquiry, Internal Logical File and External Interface File.

3.39
Internal Logical File
ILF
user recognizable group of logically related data or control information maintained within the boundary of the application being measured

NOTE An Internal Logical File is a type of base functional component.

3.40
maintain
add, change or delete data through an elementary process

3.41
meaningful
user recognizable and satisfies a Functional User Requirement

3.42
perfective maintenance
modification of a software product after delivery to detect and correct latent faults in the software product before they are manifested as failures

NOTE 1 Adapted from ISO/IEC 14764:2006, 3.7.

NOTE 2 Perfective maintenance provides enhancements for users, improvement of program documentation, and recoding to improve software performance, maintainability, or other software attributes.

NOTE 3 Contrast with: adaptive maintenance; corrective maintenance.

3.43
primary intent
intent that is first in importance

3.44
processing logic
any of the requirements specifically requested by the user to complete an elementary process such as validations, algorithms or calculations and reading or maintaining a data function

3.45
purpose of the count
reason for performing the function point count

NOTE See 5.3 a).

3.46
Record Element Type
RET
user recognizable sub-group of data element types within a data function

3.47
self-contained
no prior or subsequent processing steps are needed to initiate or complete the Functional User Requirement(s)

EXAMPLE The Functional User Requirement states that an employee is to be both added and updated. There might be multiple parts that make up the complete employee information. This can be represented by separate physical screens, windows or tabs such as

— Employee identification,

— Employee location,

— Dependent information,

— Salary information, and

— Education.

To add an employee, one or more of the tabs must be completed depending on the business rules. The add process is not self-contained until all mandatory information has been entered.

To update an employee, one or more of the tabs can be updated at any given time, but they are all process steps that meet the Functional User Requirement of updating an employee. Adding, changing or deleting information on any individual tab is not a separate elementary process. It is a process step involved in updating an employee. Even though additional information can be entered into the employee record, all of the information together is considered to be a part of the single elementary process: Update Employee.

Add Employee and Update Employee would each be a self-contained process.

3.48
sorting
activity of sequencing of rows or records in a transactional function

3.49
transactional function
elementary process that provides functionality to the user to process data

NOTE A transactional function is an External Input, External Output or External Inquiry.

3.50
user
person or thing that communicates or interacts with the software at any time

NOTE "Things" include, but are not limited to, software applications, animals, sensors and other hardware.

[ISO/IEC 14143-1:2007, 3.11]

3.51
user recognizable
requirements for processes and/or data that are agreed upon and understood by both the user(s) and software developer(s)

3.52
user view
Functional User Requirements as perceived by the user

NOTE Developers translate the user view into software in order to provide a solution.

4 Abbreviated terms

BFC Base Functional Component
DET Data Element Type
EI External Input
EIF External Interface File
EO External Output
EQ External Inquiry
FP Function Point
FPA Function Point Analysis
FTR File Type Referenced
ILF Internal Logical File
RET Record Element Type

5 Measurement Process

5.1 Overview

To conduct a function point count, the following activities shall be performed to identify and classify the base functional components (ILF, EIF, EI, EO, EQ)

a) gather the available documentation in accordance with 5.2,

b) determine the counting scope and boundary and identify Functional User Requirements in accordance with 5.3,

c) measure data functions in accordance with 5.4, 5.6 and 5.7,

NOTE Conversion functionality (if applicable) is measured in accordance with 5.6; enhancement functionality (if applicable) is measured in accordance with 5.7.

d) measure transactional functions in accordance with 5.5, 5.6 and 5.7,

NOTE Conversion functionality (if applicable) is measured in accordance with 5.6; enhancement functionality (if applicable) is measured in accordance with 5.7.

e) calculate the functional size in accordance with 5.8,

f) document the function point count in accordance with 5.9, and

g) report the result of the function point count in accordance with 5.10.

NOTE Figure 1 provides a graphical overview of the function point counting process.

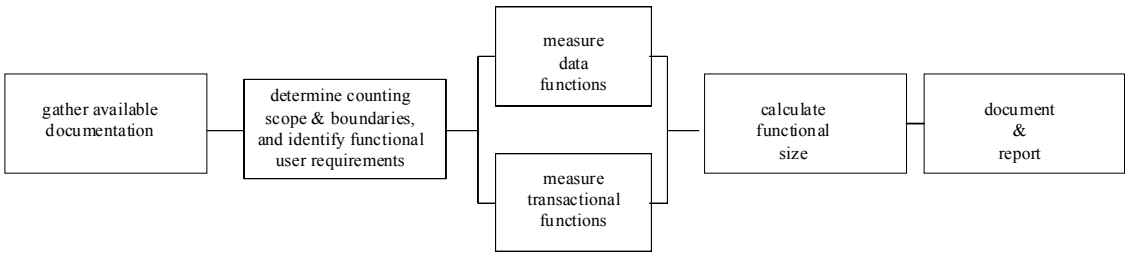

Figure 1 — Graphical overview of the function point counting process

5.2 Gather the available documentation

Documentation to support a function point count shall describe the functionality delivered by the software or the functionality that is impacted by the software project that is being measured.

Sufficient documentation to conduct the function point count or access to subject matter experts, who are able to provide additional information to address any gaps in the documentation, shall be obtained.

NOTE Suitable documentation may include requirements, data/object models, class diagrams, data flow diagrams, use cases, procedural descriptions, report layouts, screen layouts, user manuals and other software development artifacts.

5.3 Determine the counting scope and boundary and identify Functional User Requirements

To determine the counting scope and boundary of each application and identify Functional User Requirements, the following activities shall be performed

a) identify the purpose of the count,

NOTE 1 A function point count is conducted to provide an answer to a business question, and it is the business question that determines the purpose.

NOTE 2 The purpose of the count determines the counting scope.

EXAMPLE 1 The purpose of the count could be to determine the size of a particular software release.

EXAMPLE 2 The purpose of the count could be to determine the size of an application as part of the organization's effort to determine the size of its software portfolio.

b) identify the type of count, based on the purpose, as one of:

   1) a development project function point count;

   2) an application function point count;

   3) an enhancement project function point count,

c) determine the counting scope, based on the purpose and type of count,

d) determine the boundary of each application within the counting scope, based on the user view, not on technical considerations

EXAMPLE 3 If the purpose is to estimate the cost of an enhancement to the Human Resources (HR) and Benefits applications, then:

— the type of count is an enhancement project count;

— the scope shall include the transactional and data functions added, changed or deleted for both the HR and Benefits applications as well as any conversion requirements for each application;

— the user view is that HR and Benefits are different functional areas, and therefore they are separate applications;

— a boundary exists between the HR and Benefits applications as well as between each application and user.

e) the user requirements may include a mixture of both functional and non-functional requirements; identify which requirements are functional, and exclude the non-functional requirements.

5.4 Measure data functions

5.4.1 Overview

Data functionality satisfies the Functional User Requirements to store and/or reference data. All data functionality within the counting scope shall be evaluated to identify each logical group of data.

To measure data functions, the following activities shall be performed:

a) identify and group all logical data into data functions in accordance with 5.4.2,

b) classify each data function as either an ILF or an EIF in accordance with 5.4.3,

c) count the DETs for each data function in accordance with 5.4.4,

d) count the RETs for each data function in accordance with 5.4.5,

e) determine the functional complexity for each data function in accordance with 5.4.6, and

f) determine the functional size for each data function in accordance with 5.4.7.

5.4.2 Identify and group all logical data into data functions

NOTE 1 Data functions are most easily identified using a logical data model; however, this does not preclude the use of the measurement process in environments where alternative data or object modeling techniques are employed. Data modeling terminology is used to document the data function rules, but the same approach can be applied with other techniques.

To identify data functions, the following activities shall be performed:

a) identify all logically related and user recognizable data or control information within the counting scope,

b) exclude entities that are not maintained by any application,

c) group related entities that are entity dependent,

NOTE 2 Entities that are entity independent are considered to be separate logical groups of data.

d) exclude those entities below, referred to as code data:

1) substitution data entity that contains a code and an explanatory name or description;

2) single occurrence entity that contains one or more attributes that rarely change, if at all;

3) entity that contains data that is basically static or may change very rarely;

4) default values entity that contains values for populating attributes;

5) valid values entity that contains available values for selection or validation;

6) entity that contains a range of values for validation,

NOTE 3 These entities referred to above as code data may also contain other attributes for audit purposes and to define effective dates. The presence of these attributes does not alter the nature of these entities.

e) exclude entities that do not contain attributes required by the user, and

f) remove associative entities that contain additional attributes not required by the user and associative entities that contain only foreign keys; group foreign key attributes with the primary entities.

NOTE 4 Foreign key attributes are data required by the user to establish a relationship with another data function.

5.4.3 Classify each data function as either an ILF or an EIF

A data function shall be classified as an

a) Internal Logical File (ILF) if it is maintained by the application being measured, or

b) External Interface File (EIF) if it is

— referenced, but not maintained, by the application being measured, and

— identified in an ILF in one or more other applications.

5.4.4 Count DETs for each data function

To count Data Element Types (DETs) for a data function, the following activities shall be performed:

a) count one DET for each unique user recognizable, non-repeated attribute maintained in or retrieved from the data function through the execution of all elementary processes within the counting scope,

NOTE 1 For example, within an ILF or EIF, count one DET for the 12 repeated Monthly Budget Amount fields. Count one additional DET to identify the applicable month.

b) count only those DETs being used by the application being measured when two or more applications maintain and/or reference the same data function,

NOTE 2 Attributes that are not referenced by the application being measured are not counted.

c) count one DET for each attribute required by the user to establish a relationship with another data function, and

d) review related attributes to determine if they are grouped and counted as a single DET or whether they are counted as multiple DETs. Grouping will depend on how the elementary processes use the attributes within the application.

EXAMPLE The attributes (first name, middle initial, last name) are grouped and counted as

— name (first name, middle initial, last name) if these attributes are always used together,

— first names (first name and middle initial) and last name if, in addition, last name is sometimes used on its own, or

— first name, middle initial and last name, if all three can be used independently.

5.4.5 Count RETs for each data function

To count Record Element Types (RETs) for a data function, the following activities shall be performed:

a) count one RET for each data function (i.e., by default, each data function has one sub-group of DETs to be counted as one RET),

b) count one additional RET for each of the following additional logical sub-groups of DETs (within the data function) that contains more than one DET:

   1) associative entity with non-key attributes;

   2) sub-type (other than the first sub-type);

   3) attributive entity, in a relationship other than mandatory 1-1

NOTE 1 A mandatory 1-1 relationship reflects a relationship between two entities where each is related to one and only one instance of the related entity.

NOTE 2 If you don't have a data model, look for repeating groups of data in order to identify RETs.

EXAMPLE 1 An invoice has a header section of the customer information and line items of the purchases (e.g., item number, description, price, weight). The Header section is counted as one RET. The line items are a repeating group and is counted as an additional RET.

EXAMPLE 2 A single repeating attribute such as multiple account numbers for a customer is not a repeating group and is counted as a single DET and not as a RET.

EXAMPLE 3 A multiple occurring group of DETs such as year, month and budget amount is a repeating group but is counted as three DETs and not as a RET.

5.4.6 Determine the functional complexity for each data function

The functional complexity of each data function shall be determined using the number of RETs and DETs in accordance with Table 1.

Table 1 — Data function complexity

		DETs		
		1 – 19	20 – 50	> 50
RETs	1	Low	Low	Average
	2 – 5	Low	Average	High
	> 5	Average	High	High

5.4.7 Determine the functional size for each data function

The functional size of each data function shall be determined using the type and functional complexity in accordance with Table 2.

Table 2 — Data function size

		Type	
		ILF	EIF
Functional complexity	Low	7	5
	Average	10	7
	High	15	10

5.5 Measure transactional functions

5.5.1 Overview

Transaction functionality satisfies the Functional User Requirements that process data. All transactional functionality within the counting scope shall be evaluated to identify each unique elementary process.

To measure transactional functions, the following activities shall be performed:

a) identify each elementary process required by the user in accordance with 5.5.2,

b) classify each transactional function as an External Input (EI), External Output (EO) or an External Inquiry (EQ) in accordance with 5.5.3,

c) count the File Types Referenced (FTRs) for each transactional function in accordance with 5.5.4,

d) count the Data Element Types (DETs) for each transactional function in accordance with 5.5.5,

e) determine the functional complexity for each transactional function in accordance with 5.5.6, and

f) determine the functional size of each transactional function in accordance with 5.5.7.

5.5.2 Identify elementary processes

5.5.2.1 To identify each elementary process, the following activities shall be performed:

a) Compose and/or decompose the Functional User Requirements into the smallest unit of activity, which satisfies all of the following:

   1) is meaningful to the user;

   2) constitutes a complete transaction;

NOTE For users of ISO/IEC 20926:2003, item 2) is not a change, but rather a refinement to increase the specificity to promote consistent interpretation.

   3) is self-contained;

   4) leaves the business of the application being counted in a consistent state,

EXAMPLE 1 A Functional User Requirement may state that a function is to be provided to Maintain Employee information. That requirement is decomposed into smaller units of work such as Add Employee, Change Employee, Delete Employee, Inquire about Employee.

EXAMPLE 2 Individual requirements may state the need to add different types of employee information (e.g., address, salary and dependent information), but the smallest unit of activity meaningful to the user is to Add Employee.

b) Identify an elementary process for each unit of activity identified that meets all of the criteria in 5.5.2.1 a).

5.5.2.2 To determine unique elementary processes, the following activities shall be performed:

a) When compared to an elementary process already identified, count two similar elementary processes as the same elementary process if they

   1) require the same set of DETs, and

   2) require the same set of FTRs, and

   3) require the same set of processing logic to complete the elementary process (refer to the list below 5.5.2.2 b).

NOTE 1 The activities to measure transactional functions are depicted sequentially; however, they are in fact iterative. FTRs and DETs are identified in accordance with 5.5.4 and 5.5.5, but are necessary to compare two similar elementary processes.

NOTE 2 One elementary process may include minor variations in DETs or FTRs as well as multiple alternatives, variations or occurrences of processing logic below.

EXAMPLE When an elementary process to Add Employee has been identified, it is not divided into two elementary processes to account for the fact that an employee may or may not have dependents. The elementary process is still Add Employee, and there is variation in the processing logic and DETs to account for dependents.

b) Do not split an elementary process with multiple forms of processing logic into multiple elementary processes.

NOTE 3 An elementary process that accepts and validates data from the user, reads and filters data from an ILF and sorts and presents the results back to the user cannot be split into multiple elementary processes.

5.5.2.3 Various forms of processing logic to complete an elementary process are identified in Table 3.

Table 3 — Forms of processing logic

Forms of processing logic
1. Validations are performed EXAMPLE 1 When adding a new employee to an organization, the employee process validates the employee type DET.
2. Mathematical formulas and calculations are performed EXAMPLE 2 When reporting on all employees within an organization the process includes calculating the total number of salaried employees, hourly employees and all employees.
3. Equivalent values are converted EXAMPLE 3 Employee age is converted to an age range group using a table. Data is filtered and selected by using specified criteria to compare multiple sets of data;

Forms of processing logic
4. Data is filtered and selected by using specified criteria to compare multiple sets of data EXAMPLE 4 To generate a list of employees by their assignment, an elementary process compares the job number of a job assignment to select and list the appropriate employees with that assignment.
5. Conditions are analyzed to determine which are applicable EXAMPLE 5 Processing logic exercised by the elementary process when an employee is added will depend on whether an employee is paid based on salary or hours worked. The entry of DETs (and the resulting processing logic) based upon a different choice (salary or hourly) in this example is part of one elementary process.
6. One or more ILFs are updated EXAMPLE 6 When adding an employee, the elementary process updates the employee ILF to maintain the employee data.
7. One or more ILFs or EIFs are referenced EXAMPLE 7 When adding an employee, the currency EIF is referenced for the correct US dollar conversion rate to determine an employee's hourly rate.
8. Data or control information is retrieved EXAMPLE 8 To view a list of employees, employee information is retrieved from a data function.
9. Derived data is created by transforming existing data to create additional data EXAMPLE 9 To determine (derive) a patient's registration number (e.g., SMIJO01), the following data is concatenated: — the first three letters of the patient's last name (e.g., SMI for Smith); — the first two letters of the patient's first name (e.g., JO for John); — a unique two-digit sequence number (starting with 01).
10. Behavior of the application is altered EXAMPLE 10 The behavior of the elementary process of paying employees is altered when a change is made to pay them every other Friday versus on the 15th and the last day of the month; resulting in 26 pay periods per year vs. 24.
11. Prepare and present information outside the boundary EXAMPLE 11 A list of employees is formatted and displayed for the user.
12. Capability exists to accept data or control information that enters the boundary of the application EXAMPLE 12 A user enters information to add a customer order to the application.
13. Sorting or arranging a set of data. This form of processing logic does not impact the identification of the type or contribute to the uniqueness of an elementary process; i.e., the orientation of the data does not constitute uniqueness EXAMPLE 13 The list of employees is sorted in either alphabetical or location order. EXAMPLE 14 On an order entry screen, the order header information is arranged at the top of the screen, and the order details are placed below. NOTE 1 ISO/IEC 20926:2003 misstated the terms "resorting" and "rearranging"; use of the terms, "sorting" and "arranging", is a correction rather than a change.

5.5.3 Classify each elementary process as a transactional function

5.5.3.1 For each elementary process

a) the primary intent shall be identified as one of the following:

   1) altering the behavior of the application;

   2) maintaining one or more ILFs;

   3) presenting information to the user,

b) the forms of processing logic required to complete the elementary process shall be identified from the list presented in 5.5.2.3.

5.5.3.2 Each elementary process shall be classified as

a) an EI, if it:

   1) includes processing logic to accept data or control information that enters the boundary of the application;

   2) has a primary intent of either

      i) maintaining one or more ILFs, or

      ii) altering the behavior of the application,

b) an EO, if it has the primary intent of presenting information to the user and it includes at least one of the following forms of processing logic:

   1) mathematical calculations are performed;

   2) one or more ILFs are updated;

   3) derived data is created;

   4) the behavior of the application is altered,

c) an EQ, if it has the primary intent of presenting information to the user and it:

   1) references a data function to retrieve data or control information;

   2) does not satisfy the criteria to be classified as an EO.

NOTE 1 Table 4 presents a summary of the relationship between the primary intent and transactional function type.

Table 4 — Relationship between primary intent and transactional function type

Function	Transactional function type		
	EI	EO	EQ
Alter the behavior of the application	PI	F	N/A
Maintain one or more ILFs	PI	F	N/A
Present information to a user	F	PI	PI

key

PI	the primary intent of the transactional function type
F	a function of the transactional function type that is sometimes present but which is not the primary intent
N/A	the transactional function type is not allowed to perform this function type

NOTE 2 Table 5 presents a summary of the relationship between forms of processing logic and transactional function type.

Table 5 — Relationship between processing logic and transactional function type

Form of processing logic	Transactional function type		
	EI	EO	EQ
1. Validations are performed	c	c	c
2. Mathematical calculations are performed	c	m*	n
3. Equivalent values are converted	c	c	c
4. Data is filtered and selected by using specified criteria to compare multiple sets of data	c	c	c
5. Conditions are analyzed to determine which are applicable	c	c	c
6. At least one ILF is updated	m*	m*	n
7. At least one ILF or EIF is referenced	c	c	m
8. Data or control information is retrieved	c	c	m
9. Derived data is created	c	m*	n
10. Behavior of the application is altered	m*	m*	n
11. Information is prepared and then presented outside the boundary	c	m	m
12. Data or control information entering the boundary of the application is accepted	m	c	c
13. A set of data is sorted or arranged	c	c	c

key

m	it is mandatory that the transactional function type perform the form of processing logic
m*	it is mandatory that the transactional function type perform at least one of these (m*) forms of processing logic
c	the transactional function type can perform the form of processing logic, but it is not mandatory
n	the transactional function type cannot perform the form of processing logic

5.5.4 Count FTRs for each transactional function

For each transactional function, one FTR shall be counted for each unique data function that is accessed (read from and/or written to) by the transactional function.

NOTE The activities to measure transactional functions are depicted sequentially; however, they are in fact iterative. To compare two similar elementary processes, identification of FTRs is necessary as stated in 5.5.2.2.

5.5.5 Count DETs for each transactional function

To count DETs for a transactional function, the following activities shall be performed.

a) Review everything that crosses (enters and/or exits) the boundary.

b) Count one DET for each unique user recognizable, non-repeated attribute that crosses (enters and/or exits) the boundary during the processing of the transactional function.

EXAMPLE 1 DETs that cross the boundary include

— attributes the user enters via a screen as well as those displayed on a report or screen,

— attributes that enter the boundary of the application that are required to specify when, what and/or how the data is to be retrieved or generated by the elementary process,

— attributes provided by, or presented to, the user of the transactional function, and

— attributes in an electronic file that enter or exit the boundary.

c) Count only one DET per transactional function for the ability to send an application response message even if there are multiple messages.

EXAMPLE 2 If multiple error/confirmation messages are presented to the user, only one DET is counted.

d) Count only one DET per transactional function for the ability to initiate action(s) even if there are multiple means to do so.

EXAMPLE 3 If the user can initiate the generation of a report by clicking on the OK button or by pressing a function key, only one DET is counted.

e) Do not count the following items as DETs:

— literals such as report titles, screen or panel identifiers, column headings and attribute titles;

— application generated stamps such as date and time attributes;

— paging variables, page numbers and positioning information; e.g., 'Rows 37 to 54 of 211';

— navigation aids such as the ability to navigate within a list using "previous", "next", "first", "last" and their graphical equivalents;

— attributes generated within the boundary by a transactional function and saved to an ILF without exiting the boundary;

— attributes retrieved or referenced from an ILF or EIF for participation in the processing without exiting the boundary.

NOTE The activities to measure transactional functions are depicted sequentially; however, they are in fact iterative. To compare two similar elementary processes, identification of DETs is necessary as stated in 5.5.2.2.

5.5.6 Determine the functional complexity for each transactional function

The functional complexity for each transactional function shall be determined using the number of FTRs and DETs in accordance with Table 6 or Table 7.

Table 6 — EI functional complexity

		DETs		
		1 – 4	5 – 15	> 15
FTRs	0 – 1	Low	Low	Average
	2	Low	Average	High
	> 2	Average	High	High

Table 7 — EO and EQ functional complexity

		DETs		
		1 – 5	6 – 19	> 19
FTRs	0 – 1	Low	Low	Average
	2 – 3	Low	Average	High
	> 3	Average	High	High
NOTE An EQ has a minimum of 1 FTR.				

5.5.7 Determine the functional size of each transactional function

The functional size of each transactional function shall be determined using the type and functional complexity in accordance with Table 8.

Table 8 — Transactional function size

		Type		
		EI	**EO**	**EQ**
Functional complexity	Low	3	4	3
	Average	4	5	4
	High	6	7	6

5.6 Measure conversion functionality

The counting scope of a development or enhancement project can also include the functional size of conversion functionality required for the project. Conversion transactional functions and data functions (that have not already been counted) shall be counted in accordance with 5.4 and 5.5.

NOTE The counting scope dictates whether conversion functionality is counted.

5.7 Measure enhancement functionality

Enhancement projects can involve adds, changes to and deletion of existing functionality. Enhancement functionality shall be measured in accordance with the following.

a) Do not modify the boundary already established for the application(s) being modified.

b) Count data functions that are added, changed or deleted in accordance with 5.4.

c) Count transactional functions that are added, changed or deleted in accordance with 5.5.

d) The application functional size may be updated to reflect:

   1) added functionality, which increases the application functional size;

   2) changed functionality, which may increase, decrease or have no effect on the application functional size;

   3) deleted functionality, which decreases the application functional size.

NOTE 1 A change to a data function may involve adding, changing or deleting of DETs and/or RETs.

NOTE 2 A change to a transactional function may involve adding, changing or deleting of DETs and/or FTRs and/or changing of processing logic.

5.8 Calculate functional size

The purpose and counting scope shall be considered when selecting and using the appropriate formula to calculate the functional size.

A development project functional size shall be calculated using Formula (1):

$$DFP = ADD + CFP \quad (1)$$