International Standard

ISO/IEC/IEEE 32430

Software engineering — Software non-functional size measurement

Ingénierie du logiciel — Norme pour la quantification des caractéristiques non fonctionnelles des logiciels

Second edition
2025-02

Reference number
ISO/IEC/IEEE 32430:2025(en)

Institute of Electrical and Electronics Engineers, Inc
3 Park Avenue, New York
NY 10016-5997, USA

Email: stds.ipr@ieee.org
Website: www.ieee.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.

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. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC/JTC 1, *Information technology,* Subcommittee SC 7, *Software and systems engineering,* in cooperation with the Systems and Software Engineering Standards Committee of the IEEE Computer Society, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

This second edition cancels and replaces the first edition (ISO/IEC/IEEE 32430:2021), which has been technically revised.

The main changes are as follows:

— clarifications of terminology regarding software size and software non-functional requirements.

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 and www.iec.ch/national-committees.

Introduction

Used in conjunction with functional size measurement (FSM), non-functional size measurement (NFSM) assists organizations in multiple ways. It provides insight into the delivery of software projects and maintenance of software and assists in estimating the effort and in the analysis of key performance indicators, such as quality and productivity.

Having both software functional size and non-functional size provides significant information for the management of software product development. The functional size is quantifiable and represents a standardized measure of the functional project/application size. Providing a quantifiable measure derived from the non-functional requirements (NFR) for the software allows organizations to build historical data repositories that can be referenced to assist in decision making for the technical or quality aspects of applications.

By learning the method as described in this document and by performing the non-functional sizing together with functional sizing, users avoid duplication of measurement effort.

Having this information enables software professionals to do the following:

a) plan and estimate projects;

b) compare projects and compare the project to benchmarks;

c) identify areas of improvement and analyze trends of improvement;

d) quantify the impacts of the current non-functional strategies;

e) assist in determining future non-functional strategies;

f) provide specific data when communicating non-functional issues to various audiences;

g) communicate the impact of non-functional requirements (NFR) on the project with users and customers;

h) help users determine the benefit of an application package to their organization by assessing portions or categories that specifically match their requirements;

i) determine the non-functional size of a purchased application package.

NFSM is independent of the way NFR are defined. Analyzing the requirements to measure the non-functional size can assist in identifying implicit requirements.

This document contains rules on how to use ISO/IEC 20926:2009 (IFPUG FSM) and NFSM together, so that there are no gaps and no overlaps between the functional size and the non-functional size. A software requirement that contains both functional and non-functional aspects can be sized using ISO/IEC 20926:2009 for its functional aspects and this document for its non-functional aspects.

FSM and NFSM together can provide a broader view of the size of the software product.

Software engineering — Software non-functional size measurement

1 Scope

1.1 Overview

This document defines a method for measuring the non-functional size of the software. It complements ISO/IEC 20926:2009, which defines a method for measuring the functional size of the software.

This document also describes the complementarity of functional and non-functional sizes, so that deriving the sizes from the functional and the non-functional requirements does not result in duplication in the distinct functional and non-functional sizes.

In general, there are many types of non-functional requirements. Moreover, non-functional requirements and their classification evolve over time as the technology advances. This document does not intend to define the type of NFR for a given context. Users can choose ISO 25010 or any other standard for the definition of NFR. It is assumed that users size the NFR based on the definitions they use.

This document covers a subset of non-functional requirements. It is expected that, with time, the state of the art can improve and that potential future versions of this document can define an extended coverage. The ultimate goal is a version that, together with ISO/IEC 20926:2009, covers every aspect that can be required of any prospective piece of software, including aspects such as process and project directives that are hard or impossible to trace to the software's algorithm or data. The combination of functional and non-functional sizes would then correspond to the total size necessary to bring the software into existence.

Estimating the cost, effort and duration of the implementation of the NFR is outside the scope of this document.

1.2 Purpose

The purpose of this document is to define a method for measuring the non-functional size of the software.

1.3 Word usage

The word *"shall"* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted *("shall"* equals is required to).[1),2)]

The word *"should"* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*"should"* equals is recommended that).

The word *"may"* is used to indicate a course of action permissible within the limits of the standard (*"may"* equals is permitted to).

The word *"can"* is used for statements of possibility and capability, whether material, physical, or causal ("can" equals is able to).

1) The use of the word *"must"* is deprecated and shall not be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.

2) The use of *"will"* is deprecated and shall not be used when stating mandatory requirements, "will" is only used in statements of fact.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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*

ISO/IEC 20926:2009, *Software and systems engineering — Software measurement — IFPUG functional size measurement method 2009*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

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

ISO, IEC and IEEE maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

— IEEE Standards Dictionary Online: available at: http://dictionary.ieee.org

NOTE For additional terms and definitions in the field of systems and software engineering, see ISO/IEC/IEEE 24765, which is published periodically as a "snapshot" of the SEVOCAB (Systems and Software Engineering Vocabulary) database and is publicly accessible at www.computer.org/sevocab.

3.1.1
algorithm
finite ordered set of well-defined rules for the *solution* (3.1.41) of a problem

[SOURCE: ISO/IEC 2382:2015, 2121376, modified — Notes to entry have been removed.]

3.1.2
application
cohesive collection of automated procedures and data supporting a business objective, consisting of one or more components, modules, or subsystems

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

[SOURCE: ISO/IEC 20926:2009, 3.2]

3.1.3
boundary
conceptual interface between the software under study and its *users* (3.1.42)

Note 1 to entry: In this document, "application boundary" is used.

[SOURCE: ISO/IEC 14143-1:2007, 3.3, modified — Note 1 to entry has been added.]

3.1.4
code data
list data
translation data
substitutable data, which provide a list of valid values that a descriptive attribute may have

EXAMPLE Examples of code data include:

— State

  — State code

  — State name

— Payment type

  — Payment type code

  — Payment description

Note 1 to entry: Typically the attributes of the code data are code, description and/or other 'standard' attributes describing the code.

Note 2 to entry: When codes are used in the business data, it is necessary to have a means of translating to convert the code into something more recognizable to the *user* (3.1.42). In order to satisfy non-functional user requirements (e.g., usability, readability, maintainability) developers often create one or more tables containing the code data.

3.1.5
data configuration
process of adding, changing or deleting reference data or *code data* (3.1.4) information from the database or data storage with no change in software code or the database structure

3.1.6
data element type
DET
unique, *user* (3.1.42)-recognizable, non-repeated attribute

Note 1 to entry: A DET can be in business data, reference data, or *code data* (3.1.4). The number of different types of data elements of all tables is the *number of DETs* (3.1.27). Instances of control information are also counted as a DET, such as the "Enter" key to facilitate an *external input (EI)* (3.1.12).

[SOURCE: ISO/IEC 20926:2009, 3.15, modified — Note 1 to entry has been added.]

3.1.7
data function
functionality provided to the *user* (3.1.42) to meet internal or external data storage requirements

Note 1 to entry: A data function is either an *internal logical file* (3.1.21) or an *external interface file* (3.1.14).

[SOURCE: ISO/IEC 20926:2009, 3.16]

3.1.8
database view
result set of a stored query on the data, which the database *users* (3.1.42) can query just as they would in a persistent database collection object

Note 1 to entry: This stored (pre-established query) command is kept in the database dictionary. Unlike ordinary base tables in a relational database, it is a virtual table computed or collated dynamically from data in the database, when access to that view is requested. Changes applied to the data in a relevant underlying table are reflected in the data shown in subsequent invocations of the view.

3.1.9
elementary process
EP
smallest unit of activity that is meaningful to the *user* (3.1.42)

[SOURCE: ISO/IEC 20926:2009, 3.21, modified — The abbreviated term "EP" has been added.]

3.1.10
extensive mathematical operation
mathematical operation that includes one or more series of mathematical equations and calculations executed in conjunction with, or according to, logical operators, to produce results identifiable to the *user* (3.1.42)

EXAMPLE A program evaluation review technique (PERT) to calculate the expected completion date of a project.

Note 1 to entry: See also: *algorithm* (3.1.1).

3.1.11
extensive logical operation
logical operation either containing a minimum of four nesting levels, containing more than 38 *DETs* (3.1.6) required to operate the logical operation, or both

Note 1 to entry: These DETs do not necessarily have to cross the *application* (3.1.2) *boundary* (3.1.3).

3.1.12
external input
EI
elementary process (EP) (3.1.9) that processes data or control information sent from outside the *boundary* (3.1.3)

[SOURCE: ISO/IEC 20926:2009, 3.27, modified — Note 1 to entry has been removed.]

3.1.13
external inquiry
EQ
elementary process (EP) (3.1.9) that sends data or control information outside the *boundary* (3.1.3)

[SOURCE: ISO/IEC 20926:2009, 3.28, modified — Note 1 to entry has been removed.]

3.1.14
external interface file
EIF
user (3.1.42)-recognizable group of logically related data or control information, which is referenced by the *application* (3.1.2) being measured, but which is maintained within the *boundary* (3.1.3) of another application

[SOURCE: ISO/IEC 20926:2009, 3.29, modified — Note 1 to entry has been removed.]

3.1.15
external output
EO
elementary process (EP) (3.1.9) that sends data or control information outside the *boundary* (3.1.3) and includes additional processing logic beyond that of an *external inquiry* (3.1.13)

[SOURCE: ISO/IEC 20926:2009, 3.30, modified — Note 1 to entry has been removed.]

3.1.16
family of platforms
software *platforms* (3.1.29) that are serving the same purpose

EXAMPLE Operating systems; browsers.

3.1.17
file type referenced
FTR
data function (3.1.7) read or maintained by a transactional function

Note 1 to entry: A file type referenced includes an *internal logical file* (3.1.21) read or maintained by a transactional function, or an *external interface file* (3.1.14) read by a transactional function.

[SOURCE: ISO/IEC 20926:2009, 3.31, modified — Note 1 to entry has been added.]

3.1.18
function point
FP
unit of measure for functional size

[SOURCE: ISO/IEC 20926:2009, 3.35]

3.1.19
functional size measurement
FSM
process of measuring the functional size

[SOURCE: ISO/IEC 14143-1:2007, 3.7]

3.1.20
functional user requirement
FUR
sub-set of the *user* (3.1.42) requirement that describes what the software does, in terms of tasks and services

Note 1 to entry: FUR include but are not limited to:

— 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 latest aircraft position).

User requirements that are not FUR include, but are not limited to:

— 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).

[SOURCE: ISO/IEC 14143-1:2007, 3.8]

3.1.21
internal logical file
ILF
user (3.1.42)-recognizable group of logically related data or control information maintained within the *boundary* (3.1.3) of the *application* (3.1.2) being measured

[SOURCE: ISO/IEC 20926:2009, 3.39, modified — Note 1 to entry has been removed.]

3.1.22
logical file
logical group of data as seen by the *users* (3.1.42)

[SOURCE: ISO/IEC 24570:2018]

3.1.23
multiple instance approach
case where each method of delivery of the same functionality is counted separately

Note 1 to entry: See also: *single instance approach* (3.1.33).

3.1.24
non-functional size
size of the software derived by quantifying the *non-functional requirements* (3.1.25) for the software, defined by a set of rules

3.1.25
non-functional requirement
NFR
requirement for a *software-intensive system* (3.1.40) or for a *software product* (3.1.38), including how it should be developed and maintained, and how it should perform in operation, except any *functional user requirement* (3.1.20) for the software

3.1.26
NFR for software
non-functional requirements for software
portion of the *non-functional requirements (NFR)* (3.1.25) that are realized by the software

3.1.27
number of DETs
sum of all *data element types (DETs)* (3.1.6) that are part of the input/output/query of the elementary process *(EP)* (3.1.9), plus the data elements which are read or maintained internally to the *boundary* (3.1.3)

3.1.28
partition
set of software functions within an *application* (3.1.2) *boundary* (3.1.3) that share common criteria and values

EXAMPLE Partitions can be used to improve maintainability by dividing the application into two modules (within a boundary), each needing different expertise. Integrating the modules requires additional effort, which is not reflected when sizing the functional aspect of the project or product, using *function point* (3.1.18) analysis.

Note 1 to entry: Setting partitions is not based on technical or physical considerations.

Note 2 to entry: Partitions do not overlap.

3.1.29
platform
type of computer or hardware device or associated operating system, or a virtual environment, on which software can be installed or run

Note 1 to entry: Combination of an operating system and hardware that makes up the operating environment in which a program runs. A platform is distinct from the unique instances of that platform, which are typically referred to as devices or instances.

3.1.30
precautionary principle
principle that the introduction of a new product or process whose ultimate effects are unknown or disputed should be resisted until such risks are appropriately mitigated

Note 1 to entry: The precautionary principle denotes a duty to prevent harm, when it is within our power to do so, even when all the evidence is not in. This principle has been codified in several international treaties and in international law.

3.1.31
primary intent
intent that is first in importance

3.1.32
record element type
RET
user (3.1.42) recognizable sub-group of *data element types (DETs)* (3.1.6) within a *data function* (3.1.7)

[SOURCE: ISO/IEC 20926:2009, 3.46]

3.1.33
single instance approach
case where all methods of delivery of the same functionality are counted as one

Note 1 to entry: See also *multiple instance approach* (3.1.23).

3.1.34
SNAP category
software non-functional assessment process category
group of components, processes or activities that are used in order to measure the *non-functional size* (3.1.24) of the software

Note 1 to entry: Categories classify the method and are generic enough to allow for future technologies. Categories are divided into sub-categories, so that the SNAP category groups the sub-categories based on similar operations or similar types of sizing activities.

3.1.35
SNAP counting unit
SCU
software non-functional assessment process counting unit
component, process or activity, in which *non-functional size* (3.1.24) is measured

Note 1 to entry: The SCU is identified according to the nature of each sub-category. It can contain both functional and non-functional characteristics; therefore, sizing an SCU is performed for its functional sizing, using *function point* (3.1.18) analysis (FPA), and for its non-functional sizing, using SNAP.

3.1.36
SNAP point
SP
software non-functional assessment process point
unit of measurement to express the *non-functional size* (3.1.24) of the software

3.1.37
SNAP sub-category
software non-functional assessment process sub-category
component, process or activity performed within the project, to meet a *non-functional requirement* (3.1.25)

Note 1 to entry: A non-functional requirement may be sized using more than one sub-category.

3.1.38
software product
set of computer programs, procedures, and possibly associated documentation and data

Note 1 to entry: A software product is a software system viewed as the output (product) resulting from a process.

[SOURCE: ISO/IEC/IEEE 12207:2017, 3.1.54]

3.1.39
software project
set of work activities, both technical and managerial, required to satisfy the terms and conditions of a project agreement

Note 1 to entry: A software project has specific starting and ending dates, well-defined objectives and constraints, established responsibilities, a budget and a schedule. A software project may be self-contained or may be part of a larger project. In some cases, a software project may span only a portion of the software development cycle. In other cases, a software project may span many years and consist of numerous subprojects, each being a well-defined and self-contained software project.

3.1.40
software-intensive system
system for which software is a major technical challenge and is perhaps the major factor that affects system schedule, cost and risk

Note 1 to entry: In the most general case, a software-intensive system is comprised of hardware, software, people and manual procedures.

[SOURCE: IEEE 1062-2015, 3.1]

3.1.41
solution
combination of people, processes and technologies to implement a desired capability

[SOURCE: IEEE 7005-2021, 3.1]

3.1.42
user
any person or thing that communicates or interacts with the software at any time

EXAMPLE Software applications, animals, sensors, or other hardware.

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

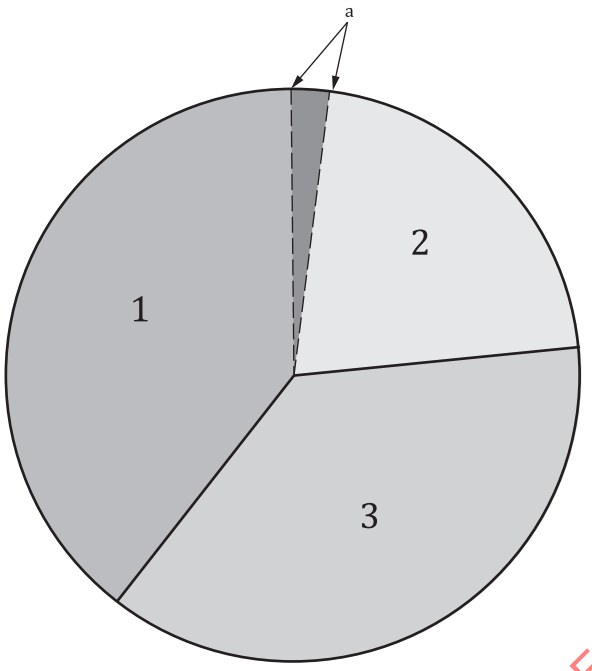
3.2 Abbreviated terms

API	application programming interface
EDI	electronic data interchange
FAQ	frequently asked questions
FPA	function point analysis
GUI	graphical user interface
HR	human resources
IATA	International Air Transport Association
IFPUG	International Function Point Users Group
NFR	non-functional requirement(s)
NFSM	non-functional size measurement
PDF	Portable Document Format
SOA	service-oriented architecture
SNAP	software non-functional assessment process
UI	user interface
WCAG	Web Content Accessibility Guidelines
XML	eXtensible Markup Language

4 Introductory information

4.1 User requirements for a system

Figure 1 illustrates how requirements can be classified.



Key

1 FUR

2 NFR for software

3 requirements for hardware and other non-software

[a] The boundary between functional requirements and NFR does not have a universally agreed definition (see 4.5.1 and 7.1). Although opinions can vary as to whether a given requirement is considered a functional or a non-functional requirement, the same requirement is not counted in both types (See Clause 7 for details).

Figure 1 — Classifying user requirements for a system

The term FUR is used to describe section 1 and is sized by an FSM. The term NFR for software, as used in this document, describes section 2 and is sized by an NFSM.

This document defines the process of measuring the non-functional size of the software, measuring the size of section 2.

4.2 Non-Functional Size Measurement (NFSM) introduction

In software engineering, size is an important attribute of software. For instance, the size of a piece of software-under-planning is used to calculate the resources (including time) necessary to develop the software, provided historical or benchmark productivity data is available. The size of a piece of software once developed is used for the normalization of certain software performance indicators (e.g., the number of defects found in the software) or for the calculation of the development organization's performance, including its productivity. This information is used to estimate the effort of future software projects.

The size of software can be measured in several ways, with different units of measure. For software that already exists, size can be measured by counting the source statements that form the software's algorithm and data. However, one major disadvantage of “source statement” as a unit of measure is that early in the development of software there are no statements yet to count. The substitute for estimating the number of source statements needed to satisfy the software requirements is quite imprecise due to a dependency on implementation specifics typically decided later in the project. This imprecision can be avoided by basing the sizing activity on present software requirements instead of future source statements.

Several sizing methods have been put forward that are based on software requirements. One of these methods is defined in ISO/IEC 20926:2009, which focuses on a subtype of the software requirements, the FUR, and assigns “function points” (FPs) (the unit of measure) to elements of FUR.

In addition to functional and non-functional size, other project-constraining attributes can be considered to provide perspective to an appropriate level of detail. This includes, for example, development methodology, the maturity level of the organization, tools, experience, quality, and collaboration effectiveness between team members.

This document complements ISO/IEC 20926:2009 by focusing on non-functional (software) requirements. Instead of FPs, this document assigns SNAP points (SPs) to NFR for software.

4.3 Software-intensive system and software product

Along with software, a software-intensive system may include hardware, data, and other components, such as documents and training. The NFR of the software-intensive system may be met by the software product or by the other components of the software-intensive system. For example, requirements for improving the response time of a system may be met by advanced hardware, by improving the efficiency of the software, or by both. NFR may be also met by the changes to the data maintained or used by the software product, and not by the software product.

4.4 Software domains

ISO/IEC 14143-1:2007 defines software functional domain as a "class of software based on the characteristics of Functional User Requirements." The term "software domains" replaces the phrase "software types" (which have been loosely defined) and is used to differentiate between categories of software products, so that performance indicators can be compared within similar types of software products.

ISO/IEC 14143-1:2007 requires that a Functional Size Measurement (FSM) Method shall describe the functional domain(s) to which it can be applied. ISO/IEC TR 14143-5:2004 defines example methods to determine the functional domain.

A non-functional size measurement (NFSM) should use the same method of functional domains, as described in ISO/IEC 14143-1:2007 when comparing the performance of the non-functional aspects of software products or software projects.

4.5 The relations between non-functional requirements (NFR) and functional user requirements (FUR)

4.5.1 Non-functional requirements

The boundary between functional requirements and NFR does not have a universally-agreed definition. NFR is defined as any requirement for the software except any functional user requirement, (i.e. excluding what the software does, but including how the software will do it).

ISO/IEC 25010 defines a product quality model and categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability, and portability). This categorization had been changed in the past and, as the software capabilities evolve, the definition of NFR and their boundary with functional requirements may change, and a new standard may supersede ISO/IEC 25010 in the future.

This document defines how to measure the non-functional size of the software irrespective of the way NFR are defined. Instead, a non-functional requirement is mapped to the category and sub-category of the software non-functional sizing method. SNAP sub-categories do not define or describe NFR; they classify only how these requirements should be sized.

4.5.2 The relations between NFR and SNAP sub-categories

The software portion of a non-functional requirement, which can be defined using ISO/IEC 25010 or other standards, shall be sized using SNAP sub-categories. SNAP measures the non-functional size of the software, and not the non-functional requirement for the system.

The categories and sub-categories and the method to size the sub-categories are defined in Clause 5. They are as follows.

a) Category 1—Data operations
    1) Sub-Category 1.1—Data entry validation
    2) Sub-Category 1.2—Logical and mathematical operations
    3) Sub-Category 1.3—Data formatting
    4) Sub-Category 1.4—Internal data movements
    5) Sub-Category 1.5—Delivering added value to users by data configuration
b) Category 2: Interface design
    1) Sub-Category 2.1—User interfaces
    2) Sub-Category 2.2—Help methods
    3) Sub-Category 2.3—Multiple input methods
    4) Sub-Category 2.4—Multiple output methods
c) Category 3: Technical environment
    1) Sub-Category 3.1—Multiple platforms
    2) Sub-Category 3.2—Database technology
    3) Sub-Category 3.3—Batch processes
d) Category 4: Architecture
    1) Sub-Category 4.1—Component-based software
    2) Sub-Category 4.2—Multiple input/output interfaces

A non-functional requirement may be implemented in the software product by using one or more SNAP sub-categories. Accordingly, a sub-category can be used to size several non-functional characteristics, as defined by this standard.

EXAMPLE A requirement to improve the way the system recovers from a crash. Using ISO/IEC 25010, this requirement falls under the area of reliability, and the sub characteristic is recoverability.

The software product is designed as follows:

— An algorithm is added to identify corrupted data in specific fields
— Time stamps are added to database records
— An algorithm is written to reconstruct corrupted data using uncorrupted records.
— The design involves the following SNAP sub-categories.
    — Database technology (adding time stamp)
    — Logical and mathematical operations

In this example, the “recoverability” type of non-functional requirement is mapped to two SNAP sub-categories, “Database Technology” and “Logical and Mathematical Operations.”

More examples are illustrated in Table 1, using characteristics from ISO/IEC 25010:2023.

Table 1 — Mapping NFR characteristics to SNAP sub-categories

		Snap categories and sub-categories														
		Data operation					Interface design				Technical environment			Archi-tecture		
ISO 25010:2023 Sub-characteristics (partial list)		Data entry validations	Logical and mathematical operations	Data formatting	Internal data movements	Delivering added value to users by data	User interfaces	Help methods	Multiple input methods	Multiple output methods	Multiple platforms	Database technology	Batch processes	Component-based software	Multiple input/output interfaces
		1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	3.1	3.2	3.3	4.1	4.2
Performance efficiency	Time Behavior											Ex. 3			
	Resource utilization														
	Capacity														
Interaction capability	Appropriateness recognizability														
	Learnability						Ex. 1								
	Operability								Ex. 4						
	User error protection														
	User engagement														
	Inclusivity														
	User assistance						Ex. 1								
	Self-descriptiveness														
Reliability	Faultlessness														
	Availability														
	Fault Tolerance														
Security	Confidentiality														
	Integrity														
	Non-repudiation														
	Accountability														
	Authenticity	Ex. 2													
	Resistance														
Maintainability	Modularity														
	Reusability														
	Analysability														
	modifiability														
	Testability														
Safety	Operational constraint														
	Risk identification														
	Fail safe														
	Hazard warning														
	Safe integration														

EXAMPLE 1 Improving the user assistance and learnability by adding pop-up help menus (Sub-category 2.1) and rearranging the screens (Sub-category 2.2).

EXAMPLE 2 Improving security by adding more validations to the authentication process, using Sub-category 1.1 “Data Entry Validations” and Sub-category 1.2 “Logical and Mathematical Operations.”

EXAMPLE 3 Improving performance by adding indices to the database and improving queries (Sub-category 3.2).

EXAMPLE 4 Adding barcode reading as an additional input method (Sub-category 2.3).

4.6 Current classification and Future evolution of NFR

4.6.1 The challenge

As systems evolve, new variations on the types of requirements emerge from the enhanced capabilities of software. Machine learning, artificial intelligence, and the Internet of Things generate the need for new types of requirements, such as ethics, social responsibility, privacy, and sustainability. Similar to other NFR, such as security and accountability, these attributes shape the functional behaviours of the system and its components.

These requirements may fail to gain sufficient notice by software development teams until after a user finds a fault or a logic error or the system in use has had unintended consequences on user communities. Then everyone takes notice, with software development teams working to find corrective and preventive mitigations to address what may not fit the current definitions of functional requirements and the classification of NFR.

Similar to other requirements that are intended to shape system behaviours through the assurance of adherence to defined values, emerging requirements should reflect stakeholder values, some of which are often not explicitly defined or beyond the immediate access of the development environment (e.g., robotics as prosthetics). Software development teams need to understand the system’s context of use, the risks of harm, then the application of the “precautionary principle” as the risk mitigation technique necessary to anticipate and mitigate against undesirable consequences such as harm to life of any kind and quality of life, the environment, a species, or a community.

4.6.2 Current classification of NFR

Emerging social requirements fall within the category of quality in use:

ISO/IEC 25019:2023 defines quality in use as the extent to which the system or product, when it is used in a specified context of use, satisfies or exceeds stakeholders’ needs to achieve specified beneficial goals or outcomes[9]. The quality in use of a system characterizes the impact that the product (system or software product) has on stakeholders. It is determined by the quality of the software, hardware, and operating environment, and the characteristics of the users, tasks, and social environment. All these factors contribute to the quality in use of the system.

Systems should be evaluated by the following criteria.

a) How risk-free is the use of the system?

b) How risk-free does updating the content of the system need to be?

c) How risk-free does making maintenance changes to the system or porting the system need to be?

d) How risk-free does using the output from the system need to be?

4.6.3 Sizing quality-in-use requirements

Meeting these user requirements consumes time, effort, and other resources that should be evaluated and considered when planning and managing a software project. Therefore, like other user requirements, they should be sized.

SNAP can be used to size quality-in-use requirements. For example, algorithms to analyze if a certain situation may impose risk, and assess the probability and the impact of such a risk. This can be sized using Sub-category 1.2—Logical and Mathematical Operations. Storing data for this purpose may be sized using

Sub-category 3.2—Database Technology. However, the current set of sub-categories may be enhanced in the future to cover more aspects of these new requirements.

4.7 Objectives and benefits

4.7.1 Objectives

SNAP measures the non-functional size of the software derived by quantifying the Non-Functional Requirements. The objectives of SNAP are to:

a) Measure the non-functional size of the software that the user requests and receives;

b) Demonstrate the full economic value of the application, including its functional aspects as well as the non-functional aspects (have the non-functional application size in addition to the functional application size, to demonstrate the full economic value);

c) Measure software development and maintenance projects based on the NFR for the software;

d) Size technical projects, to which FPA is not applicable.

In addition to meeting the above objectives, the process of measuring the Non-functional size of the software should be:

a) Simple enough to reduce the overhead of the measurement process;

b) A consistent measure among various projects and organizations.

SNAP allows determining (by counting each of the four categories, from each one of the sub-characteristics) the possibility of sizing and, therefore, better estimating a project with/without FPs, according to the set of user requirements received for a project.

4.7.2 Benefits

Measuring the non-functional size of the software assists organizations in multiple ways. It provides insight into the delivery of projects and maintenance of applications to assist in estimating and in the analysis of quality and productivity. Used in conjunction with FSM, the non-functional size provides information that can identify items impacting quality and productivity positively or negatively.

Having this information enables software professionals to:

a) Plan and estimate projects;

b) Identify areas of process improvement;

c) Assist in determining future non-functional strategies;

d) Quantify the impacts of the current non-functional strategies;

e) Provide specific data when communicating non-functional issues to various audiences.

Organizations can apply SNAP as:

a) A method to measure;

b) A method to estimate the cost and resources required for software development and maintenance;

c) A method to measure cost reduction for software development and maintenance, in addition to FPA;

d) A normalization factor for software comparison;

e) A method to determine the non-functional size of a purchased application package by sizing all the portions and categories included in the package;

f) A method to help users determine the benefit of an application package to their organization by sizing portions or categories that specifically match their requirements;

NOTE 1 The equation "(FPs + SPs)" are not equal to the overall product size. See next paragraph.

In this document, the size of a software application is considered to have two distinct parts: the size of the FURs and the size of the NFRs for the software. For example, if an application's functional size is 700 FPs and the non-functional size is 200 SPs, then the entire size can be stated as "700 FPs and 200 SPs." The two sizes do not sum up to "900 points."

The ISO/IEC 20926:2009 FSM (IFPUG functional size methodology) does not change when measuring the non-functional size using SNAP.

NOTE 2 A project can have zero function points and a nonzero number of SPs, or zero SPs and a nonzero number of function points, or any combination of function points and SPs.

5 Non-functional size: Categories and sub-categories

5.1 Category 1: Data operations

The data operations category relates to how data is processed within the SNAP counting unit (SCU) to meet the NFR in the application.

5.1.1 Sub-category 1.1: Data entry validation

5.1.1.1 SNAP Counting Unit (SCU)

The SCU is the elementary process (EP).

5.1.1.2 Vocable

Data entry validations are performed either to allow only validated (predefined) data or to prevent the acceptance of unvalidated data.

The nesting level is the number of conditional validations (If-Else combo/"While" loop/"For" loop or any other validation blocks) in the longest chain of validation.

5.1.1.3 Complexity parameters

The nesting level complexity is shown in Table 2.

Table 2 — Nesting level complexity

Nesting level complexity	Condition
Low complexity	2 nesting levels or fewer
Average complexity	3 to 5 nesting levels
High complexity	6 nesting levels or more

The number of unique DETs used across all validations shall be counted. For example, there are two fields validated in the SCU, one uses three DETs for nested validation and another uses one DET, which is not one of the three DETs above, count four DETs.

5.1.1.4 SNAP Points (SPs) Calculation

Identify the complexity based on nesting level (Table 3). Calculate SPs based on the constant factor and the number of DETs (#DETs).

Table 3 — SNAP sizing for data entry validations

Nesting level complexity	Low	Average	High
SP	2 × the number of DETs	3 × the number of DETs	4 × the number of DETs

Data entry may result from any source (e.g., UI, transactions).

The number of nesting levels is derived from the business requirements and the solution, and not from how the code is written.

Validations are nested when there is a dependency between validations.

EXAMPLE 1 A number must be between 0 and 10: if it is less than 1, it must have two digits after the decimal point. If it is 1 or more, it can have one or no digits after the decimal point.

Several validations on a field are not nested when they are independent.

EXAMPLE 2 A value must be numerical, greater than 0 and smaller than 1,000,000.

This sub-category may include requirements for error handling or exception handling.

DETs refer to all types of data.

If code data is used for data entry validations, then any changes to code data values (add/change/delete) is counted using this category.

EXAMPLE 3 A date field must have a certain size; a value entered must be in a certain range of values; a code must be present in a certain table; a field is bound to the value of the previous field (e.g., state, county, city).

EXAMPLE 4 Data entry validation enabled using code data for validation of airport names. A traveling order application has a screen with details of the departure airport, destination airport and the option to add multiple destinations. The current system validates the airport abbreviations (such as LHR, NYC) but cannot identify the airport name. In this example, the user requirement is that the user shall be able to key in either the abbreviation or the airport name. The design is to use existing code data with all airports and IATA airport codes, and to add validation rules both for the airport abbreviation and airport name. Three EPs were identified (order a flight, amend order, cancel order) using this code data validation. One nesting level and one DET are used for SNAP counting.

EXAMPLE 5 Data entry validation enabled using logical checks on the DETs—adding an employee to an organization. In the processing logic of adding an employee to an organization, the number of validations performed during data entry on a screen and the complexity level of these validations, are considered non-functional. Employee ID, in this example, is not generated automatically but manually entered besides employee name, department, date of birth and more. Validating that the set of data is correct is sized using this subcategory. (These are considered to be technical requirements.)

5.1.2 Sub-category 1.2: Logical and mathematical operations

5.1.2.1 Extensive mathematical operations

SNAP defines an "extensive mathematical operation" as a mathematical operation that includes one or more algorithms. Examples of extensive mathematical operations include using the Program Evaluation Review Technique (PERT) to calculate the expected completion date of a project, calculating the optimal profit for a business process using linear programming, determining the way to formulate the fastest flowing waiting lines using queuing theory, and finding the shortest route through a network. Examples of other algorithmic mathematical operations fitting the definition of "extensive" include solving calculus integration formulas, calculating federal income taxes, GPS calculations, gaming, weather forecasting, and perhaps calculating retirement pensions.

The DETs counted are the set of those required to operate the extensive mathematical operation, such as values for an algorithm's variables and settings maintained by the algorithm's control information. These values and settings are not necessarily stored in a single physical file; they may be stored in various locations such as settings of the value of variables located in the code or as DETs in various physical files. Regardless of how they are located, as a set(s), this satisfies the requirements for either an internal logical file(s) or external interface file(s) because they are the logical grouping(s) of data necessary to operate the algorithm.

"Simple" or "routine" mathematical operations are defined here as algorithms that are not considered as extensive operations. Examples:

— Adding a column of numbers.

— Balancing a checking account.

— Totalling daily sales.

— Calculating averages.

Iterated simple mathematical operations should not be considered extensive. For example, a fast-food restaurant manager may need to place an order for ketchup packets from a supplier. The manager first counts the current inventory of ketchup packets, forecasts the expected usage, and places an order to make up for the expected resulting shortfall. If the manager has 100 types of items in inventory and must perform this calculation 100 times to complete the total order with the supplier, then this is still defined as being a simple or routine mathematical operation because the simple or routine mathematical operation is iterated 100 times.

Sizing an EP using ISO/IEC 20926 is determined by the type of processing logic used by the external input (EI), external output (EO), or external inquiry (EQ). While this can give a higher size to the EP that contains mathematical operations, it does not necessarily correlate to the effort needed to produce extensive mathematical operations, when non-functional requirements are involved (for example, processing logic to deliver reliability, or security, or accuracy). SNAP size compensates for the additional complexity of extensive mathematical operations.

5.1.2.2 Extensive logical operations

The outcome of a logical operation may be a decision or set of decisions or evaluating a condition using data that exists in one or more logical files. The SCU is the EP.

If more than one logical operation can be executed within the EP, then count either the combined number of DETs in the operations containing a minimum or four nesting levels, or the sum of the DETs involved in all of the logical operations, provided that the sum is more than 38 (whichever is larger).

5.1.2.3 SNAP Counting Unit (SCU)

The SNAP counting unit (SCU)is the elementary process (EP).

5.1.2.4 Complexity parameters

SCU complexity parameters are as follows.

a) Number of FTRs being accessed to do the business logic processing (Table 4).

Table 4 — Number of FTRs, logical and mathematical operations

Number of FTRs	0–3 FTRs	4-9 FTRs	10+ FTRs
Complexity Level	Low	Average	High

b) Processing logic type of the EP (logical/mathematical) (Table 5).

Table 5 — EP type for logical and mathematical operations

EP Type	Main Purpose of the EP	Example
Logical	Decision-making or evaluating a condition using data that exists in one or more logical files (internal or external)	Exception processing
Mathematical	Transformation of data or use of control information that exists in one or more logical files (internal or external) that is used for an extensive mathematical operation.	Complex tax calculation

c) Number of data element types (#DETs).

5.1.2.5 SNAP Points (SPs) Calculation

Identify the complexity based on the number of FTRs and the processing logic type of the EP. Calculate size based on the number of FTR, the EP type, and the number of DETs (Table 6).

Table 6 — SNAP sizing for logical and mathematical operations

Complexity Level	Low	Average	High
SP per Logical EP type	4 × the number of DETs	6 × the number of DETs	10 × the number of DETs
SP per Mathematical EP type	3 × the number of DETs	4 × the number of DETs	7 × the number of DETs

When the main purpose of the EP cannot be clearly identified, select "Logical" (do not count it as one logical and one mathematical).

EXAMPLES Extensive logical and mathematical operations include the following:

a) Project scheduling critical path analysis

b) Complex tax calculations

c) Linear programming algorithms

d) Calculus integration formulas

e) Financial return on investment calculations for a large industrial machine

f) Statistical analysis of variance calculations

g) Business sales forecasting using the ensemble forecasting method

5.1.3 Sub-category 1.3: Data formatting

Data formatting is a requirement that deals with the structure, format, or administrative information in a transaction not directly relevant to functionality that is seen by the user.

5.1.3.1 SNAP Counting Unit (SCU)

The SCU is the EP.

5.1.3.2 Complexity parameters

Sub-category 1.3 complexity parameters include the following:

a) Transformation complexity is low, average, or high.

   1) Low: Data type conversions or simple formatting such as byte padding, or data substitution, using a maximum of 2 operators (Celsius to Fahrenheit, single integer to double integer).

   2) Average: Involves encryption/decryption, which is a characteristic of the application and applies to almost all processes that is provided through a library—API interface.

   3) High: Involves local Encryption/Decryption.

b) Number of data element types (#DETs) transformed.

5.1.3.3 SNAP Points (SPs) Calculation

Identify the complexity based on transformation. Calculate SPs based on the constant factor and the number of DETs (#DETs) (Table 7).

Table 7 — SNAP sizing for data formatting

Transformation complexity	Low	Average	High
SP	2 × the number of DETs	3 × the number of DETs	5 × the number of DETs

Data elements refer to all types of data.

The encryption algorithm is complex for:

a) Design specifically allowed several key lengths;

b) Providing a method to check data integrity for high-volume data;

c) Formatting medical images;

d) Restructuring huge volume database.

EXAMPLE 1 Simple transformations:

a) Converting text to value or value to text; converting strings to values.

b) Data formatting required for reporting requirements.

c) An application shows the date (MMDDYYYY), time - Greenwich Mean Time (GMT), current atmospheric temperature (degrees Fahrenheit) in a standard format. However, due to regulations, the date is required to be displayed as 'YYYYMMDD, the time should always show the local time zone, and temperature should be displayed as "Degrees Kelvin."

d) The display formats must be converted to adhere to the standards prescribed.

EXAMPLE 2 Complex transformations:

a) Enabling multi-lingual support for an application by using code data.

b) Encryption/ decryption, compression–decompression.

c) Compliance with standards for the electronic transfer of data in a healthcare environment. The data packets are sent and received in a particular format called electronic data interchange (EDI). For example: change the structure of the transactions (add headers and footers); the transaction format is changed per HIPAA (Health Insurance Portability and Accountability Act of 1996) requirements with no changes in the functionality.

d) Data interchange formats—XML to other formats, or other means of data interchange between two computer systems.

e) Preparation of metadata for various screen requirements or data warehouse views.

f) Transformations in data warehouse.

5.1.4 Sub-category 1.4: Internal data movements

Internal data movements are from one partition to another within application boundary with specific data handling.

5.1.4.1 SNAP Counting Unit (SCU)

The SCU is the portion of the EP that crosses from one partition to another.

The SCU is identified by the EP and the two partitions crossed. If an EP crosses more than two partitions, use the formula below per each partition crossing [in Figure 2, an EP may move from Component 1 to Component 2 (labelled "A"), and then to Component 3 (labelled "B"). In such a case, SPs shall be calculated at each partition crossing].

While ISO/IEC 20926:2009 (IFPUG FSM) refers to transactions that cross the application boundary—for Sub-category 1.4 SCU—this document refers to their internal processes/functions, which move from one partition to another.

5.1.4.2 Complexity parameters

Sub-category 1.4 complexity parameters include the following.

a) The number of unique data element types (#DETs) that are transferred from one partition to the other and processed or maintained.

b) The number of unique FTRs either read or maintained by the EP at both partitions crossed.

5.1.4.3 SNAP Points (SPs) Calculation

Identify the complexity level based on the number of FTRs read/updated and the number of DETs transferred. Calculate size as per Table 8 for each partition crossing.

Table 8 — SNAP sizing for internal data movements

Complexity Level	Low	Average	High
FTR	0–3 FTR	4-9 FTR	10+ FTR
SP	4 × the number of DETs	6 × the number of DETs	10 × the number of DETs

Internal data movements sub-category sizes internal transactions within the boundary of an application. These transactions are sized in case they cross partitions.

Any data function that crosses partitions generates SPs in addition to FPs.

When an EP crosses the partition in both directions, SNAP is sized as follows:

a) One SCU, if the transactions are synchronous;

b) Two separate SCUs, if the transactions are asynchronous.

EXAMPLES The following are examples of EPs that can have data crossing partitions:

a) Data backup within application boundary, crossing partitions.

b) Data copy/movement between tables within application boundary, crossing partitions.

c) Realign data in temporary storage.

d) Transactions between application and middleware that do not cross functional boundaries.

e) Service-oriented architecture (SOA) solutions. (When SOA functionality are within the application boundary.)

f) Data movements between tiers that do not cross-functional boundaries.

g) Data formatting transactions that use data that crosses partitions.

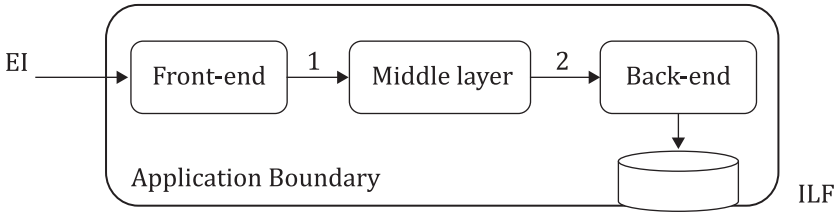
h) Internal transactions for logical /mathematical operations.

i) Reference data maintenance.

Sub-category 3.3 (Batch Processes) covers batch jobs. Batch jobs may be executed within a single partition. If the batch job crosses partitions, then it may need to be sized for crossing partitions by this sub-category.

EXAMPLE

An EP “process invoice” has two partition crossings (Figure 2).



Key

Partition 1 front-end

Partition 2 middle layer

Partition 3 back-end

1 and 2 designate the internal transaction"

Figure 2 — Example: Internal data movement

Each component references/updates a unique set of FTRs during the processing as follows:

a) Partition 1—2 FTRs

b) Partition 2—4 FTRs

c) Partition 3—3 FTRs

For this process six DETs are crossing from Partition 1 to Partition 2 and five (5) DETs are crossing from Partition 2 to Partition 3.

For the SCU “1 crossing” (from front-end to middle layer):

Number of FTRs = 2 + 4 = 6 (Average complexity)

Number of DETs = 6

SP = 6 × the number of DETs = 36

For the SCU “2 crossing” (from middle layer to back-end):

Number of FTRs = 4 + 3 = 7 (Average complexity)

Number of DETs = 5

SP = 6 × the number of DETs = 30

5.1.5 Sub-category 1.5: Delivering added value to users by data configuration

Delivering additional unique business value to users by data configuration.

5.1.5.1 SNAP Counting Unit (SCU)

The SCU is the EP per logical file.

The SCU is the EP that is created or changed because of the changes done to the logical file (and provides the added value to the users).

The EP to create the reference data or code data as defined above shall not be considered under this sub-category

EXAMPLE A new service is defined by adding its attributes to reference tables. The application is flexible enough to provide the new service with no code changes. EPs to be counted may be: add the new service; change this service; cease the service. The process to add the service attributes to the reference tables (i.e., writing and using scripts) is not counted.

When more than one EP uses the configured data, count each EP separately.

5.1.5.2 Vocable

5.1.5.2.1 Attribute

An attribute is an independent parameter that has a unique business meaning and contains a set of different values.

5.1.5.2.2 Record

A record is one row in a logical file.

5.1.5.2.3 Complexity parameters

The complexity parameters for Sub-category 1.5 are as follows:

— Number of unique attributes involved in the EP, that are added/modified/deleted.

— Number of records configured.

5.1.5.3 SNAP Points (SPs) Calculation

Identify the complexity level based on #Records. Calculate size based on the constant factor and the #Attributes (Table 9).

Table 9 — SNAP Sizing for delivering added value by data configuration

Complexity level	Low	Average	High
Records	1–10 records	11-29 records	30+ records
SP	6 × the number of attributes	8 × the number of attributes	12 × the number of attributes

NOTE 1 New services, products, or price plans can be added to the application by adding or changing reference data and not by writing code.

NOTE 2 Functionality by data configuration brings added value to the user, also adds effort to configure and test the new functionality.

Examples of delivering added value to users by data configuration are as follows:

EXAMPLE 1 An application requires granting access to a specific role in the application. To meet this requirement, the developer does not write any separate code and instead updates a configuration file, and associates the user or set of users into some property file(s). Such additions or changes are made to meet user requirements, which affect the functionality at the EP level.

The process to configure the data in the database is not sized separately. Only the user's processes are counted.

EXAMPLE 2 An application requires configuring a new product ("Product X" here below) or a component that can be sold using the application. The new product and its price plan are defined in reference data. The project effort can be creating the data, by migrating it to the reference files and testing that the application functions with the new data. The sizing process identifies many SCUs here.

a) Change product Y to product X.

b) Provide a new product X.

c) Change price of product X.

5.2 Category 2: Interface design

The interface design category relates to the end-user experience. This category assesses the design of graphical UI processes and methods that allow the user to interface with the application.

5.2.1 Sub-category 2.1—User interfaces

Unique, user identifiable, independent graphical user interface elements can be added or configured on the user interface that do not change the functionality of the system but affect non-functional characteristics (such as usability, ease of learning, attractiveness, and accessibility).

5.2.1.1 SNAP Counting Unit (SCU)

The SCU is the set of screens as defined by the EP.

5.2.1.2 Vocable

a) UI element (user interface element) is a unique user identifiable structural element that makes up the user interface. It includes elements such as:

   1) Window (which can be container, child, text, or message window);
   2) Menus;
   3) Icons;
   4) Controls;
   5) Pointer (or mouse cursor);
   6) Text box;
   7) Button;
   8) Hyperlink;
   9) Drop-down list;
   10) List box;
   11) Combo box;
   12) Checkbox;
   13) Radio button;
   14) Cycle button;
   15) Data grid;
   16) Tabs;
   17) Interaction elements like a cursor;
   18) Labels.

The above elements are used to display or manipulate data objects. The aspect that adds to the complexity of the design, configuration and testing time of a user interface is the configuration of each of these elements. NFR for the software may involve changing the properties of these UI elements. Depending upon the type of the user-interface element, varying number of properties can be configured to produce the desired output.

For example, button can be set to “locked” or “highlighted” or “coloured” or placed at a particular location on the screen.

b) UI element properties: Each UI element is associated with certain properties, which define the behaviour and look and feel of the user interface element.

c) UI element set: A UI element set is the collection of all the UI elements of the same type in the SCU.

EXAMPLE The following examples come from WCAG clause 18:

a) A window can have properties like background colour, active border, and active caption.

b) A button can have properties like button highlight, button text, and background colour.

c) Tool tips can have properties like info text, and info background.

EXAMPLE All the text boxes in the set of screens (SCU).

5.2.1.3 Complexity parameters

The complexity parameters for Sub-category 2.1 include the following:

a) The sum of the number of unique properties configured for each UI element in the SCU;

b) The number of unique UI elements affected (added, changed, or deleted).

5.2.1.4 SNAP Points (SPs) Calculation

Identify the complexity based on the number of properties of UI element set. Calculate size as the product of the constant factor and the number of unique UI elements (Table 10).

Table 10 — SNAP sizing for user interface

UI Type complexity	Low	Average	High
Number of properties	< 10 properties added or configured	10 to 15 properties added or configured	16+ properties added or configured.
SP	2 × the number of unique UI elements	3 × the number of unique UI elements	4 × the number of unique UI elements

5.2.1.5 Rules

The following rules for Sub-category 2.1 apply.

If the EP for adding/changing of UI is FP counted, then do not duplicate the count; however, changing the contents and appearance of a UI element needs to be sized as non-functional.

Aesthetic changes in UI screen, static or dynamic UI pages, rearranging of the screen and printed reports without changing their functionality shall be sized under user interfaces sub-category.

The sets of screens within one process shall be counted as the SCU.

UI elements added may be of any form, such as text, sound, picture, logos, colour, keys, controls, navigation, animating or enabling/disabling the above (when such enabling/disabling is not functional).

This subcategory may include added operations that support function keys, auto fill, shortcut keys, common keys, and navigation screen/page level.

Screens that are not considered functional (such as Administrator’s screens) and hence are not counted using FPs, are counted by SNAP using 2.1—User Interfaces Sub-category.

EXAMPLE

Some text on users' screens is hard coded. Due to a new policy of the company, the request is to replace the word "customer" with the words: "business partner."

The analysis found that the word "customer" must be replaced in the following UI elements (note that since SNAP counts the number of unique UI elements, there is no need to estimate the number of occurrences of each unique UI element).

a) SCU 1: Acquire a new business partner: header, labels, radio button, drop-down list.

b) SCU 2: Modify details of a business partner: header, labels.

c) SCU 3: Send a message to a business partner: header, labels.

d) SCU 4: Cease services to a business partner: header, labels.

Changing the text in these UI elements is considered one property. UI type complexity is low.

SP = 2 × the number of unique UI elements per each SCU:

SP = 2 × (4 + 2 + 2 + 2) = 20 SP

5.2.2 Sub-category 2.2—Help methods

Help is information for users that explains how the software provides its functionality or other supportive information provided to users.

5.2.2.1 SNAP Counting Unit (SCU)

The SCU is the Help Object.

5.2.2.2 Vocable

a) A Help Object is a particular part of the software for which the Help Item is provided.

b) A Help Item is the smallest, unique, use- identifiable information topic, which provides the user with supportive information or details about a particular part of the software. See Table 11.

EXAMPLES A DET in an EP, a report, a static web page, an icon or a picture on a screen.

Table 11 — Examples of Help Objects and Help Items

Help Object	Help Item example
A report	A framed text explaining how to interpret a report, which is displayed by pressing a "?" mark near the header of the report
A DET (e.g., an attribute entered or displayed, a control)	A pop-up text box, which is displayed when hovering over a field
A menu item	A pop-up text box, which is displayed when hovering over a field
A static web page	The static web page (a static web page is one that is delivered to all the users exactly as stored, displaying the same information to all users, and is not generated by an application).
A link to an explanation page (e.g., to a "Change password" screen, explaining how to select a strong password)	A Section in the "Help" window, explaining the rules for a password strength
A screen	A framed text explaining the purpose of the EI
A link or an icon to open a multimedia item such as a voice message or a movie[a]	A movie
An FAQ	The FAQ text

[a] SNAP does not refer to the size and the costs to produce the multimedia item, only the size for addressing those Help Items in the software.

5.2.2.3 Complexity parameters

The inclusion of screenshots in the Help Item is a complexity parameter.

5.2.2.4 SNAP Points (SPs) Calculation

Count the number of Help Objects and divide by 16 to calculate the SPs. However, if there is at least one screen shot accompanying the Help Object, count the number of Help Objects, divide by 16, and add 2 (Table 12).

Table 12 — SNAP sizing for Help Items

	With no screenshots	With at least one screenshot
SP =	(# of Help Objects)/ 16	[(# of Help Objects)/16] + 2

The media type of the Help Item is immaterial for SNAP. For example, information provided to the user may be provided by a drop-down, static web page, paper, or other media type. However, if the referenced Help Objects are the same, then the SNAP size of each media type is the same.

Do not count multiple instances of the same Help Items, regardless of media type, with the exception of a general user Help Manual. A general user Help Manual is countable in itself.

Do not add SNAP size for Help that is a help data function per ISO/IEC 20926:2009 (IFPUG FSM).

This document does not currently attempt to size help delivered by a person or persons (i.e., help that is not part of the automated software being delivered). This includes phone calls to a help desk, chat rooms, help provided by online videos, or similar products. It does not address help for an index, glossary, or table of contents.

Figure 3 is a hypothetical screen representing an EI for students registering for a university summer program. Three examples of help are identified for this screen (EI) as described in examples 1, 2, and 3. The following examples refer to Figure 3.

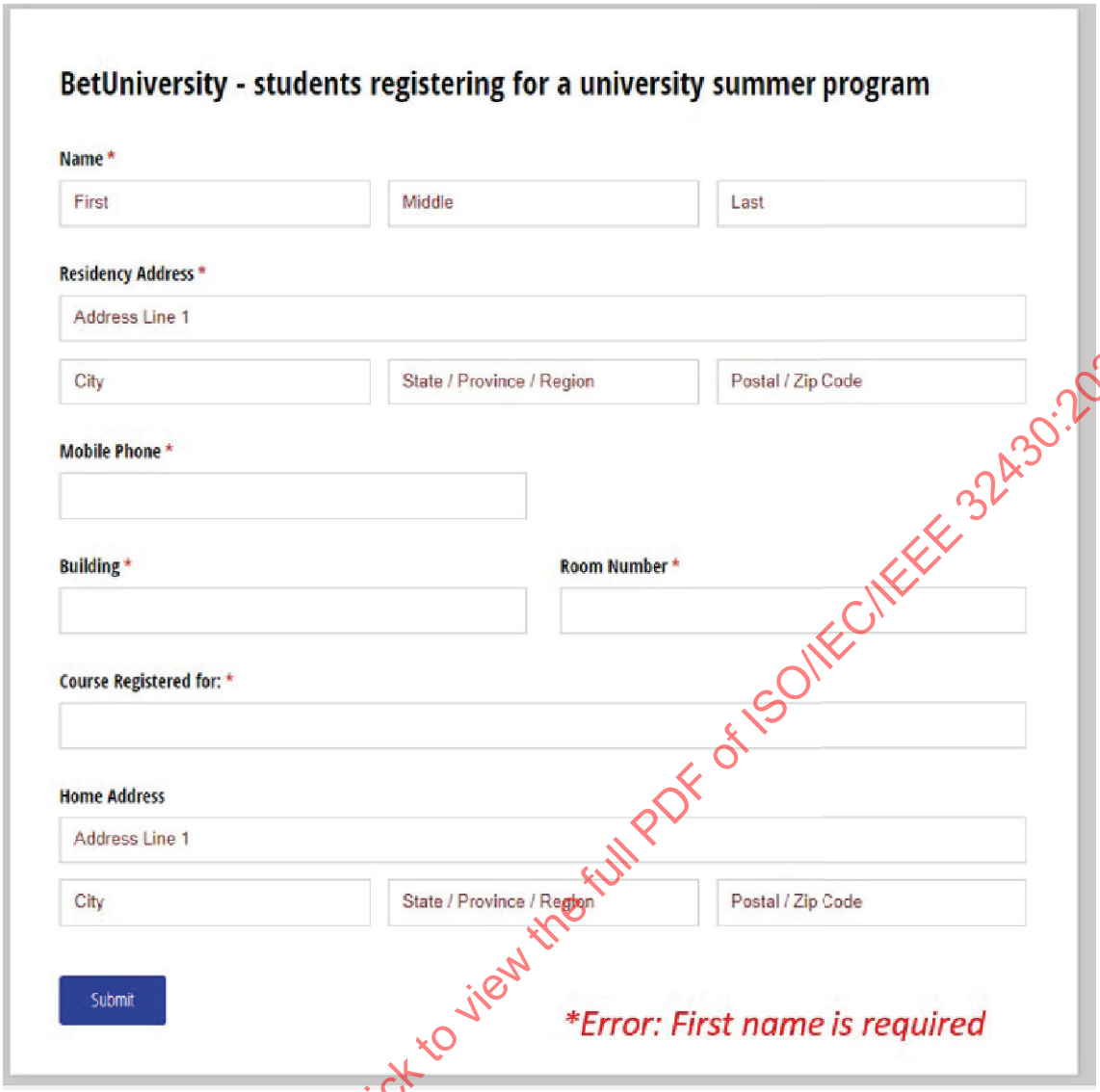


Figure 3 — A screen example

EXAMPLE 1 Screen-level help. Suppose that help is provided describing the purpose of the screen as a whole but does not go into detail about the meaning of the individual fields (or DETs). The screen is the "Help Object" and the information provided, which explains the purpose of the screen is the "Help Item." The non-functional size of the help is based on the number of Help Objects affected. There is one "Help Object" in this example.

EXAMPLE 2 Pop-up text box help. A pop-up text box is displayed when hovering over each of the fields that provided information about the meaning of the corresponding field. This includes the Submit key and the ERROR MESSAGE DET. In this situation, each DET is a Help Object and each pop-up text box is an individual Help Item. If the user recognizes 17 DETs here, there are 17 Help Objects for this example.

EXAMPLE 3 User manual help. A user manual provides information on the meaning of the 17 DETs; this includes the Submit key and the ERROR MESSAGE DET. In addition, a screenshot of this input screen is included in the User Manual to aid in the explanation. In this situation, count 17 Help Objects and count one screenshot. A screenshot was also found to correlate with work effort to develop help.

EXAMPLE 4 Help for other SNAP subcategories. Under SNAP SC 1.2, Logical and Mathematical Operations, SNAP recognizes an algorithm containing 16 countable DETs under SNAP rules. Help is provided which describes the theory of this algorithm in detail including how the algorithm uses each of these16 DETs. Count 16 Help Objects for the algorithm. Suppose this help is provided as several pages of text in a user's manual. Then this text is one Help Item addressing 16 Help Objects.

EXAMPLE 5 Menu help. Pop-up balloon help is provided for buttons at the top of a word processor screen, such as the “Bold” button or the “Italics” button. Count one Help Object for each button having balloon help and count 1/16 SP for each Help Object.

EXAMPLE 6 Login screen help. An application has a typical login screen that requires a username, password, and other fields of data, countable using FPs as five DETs. A Help Item is provided in the user manual, which describes the general purpose of the login screen but does not provide any significant detail on what each of the DETs means or about the data validation logic incorporated in the login process. Count one Help Object for the login screen as a whole, and 1/16 SP.

EXAMPLE 7 Help in the form of a video. An icon is added to a login screen as described in Example 6, in which an instructor explains the general purpose of the screen and how to fill the 5 fields. Count 1 Help Object for the general explanation and 5 Help Objects for the explained fields, which provides 6/16 SPs.

NOTE There are many cases in which adding a Help Item involves UI effort. In such cases, it is not expected to size this activity twice, as SPs for help sub-category and additional SPs for user interfaces. When the primary intent of the activity is creating a Help Item, only the help methods sub-category is counted.

5.2.3 Sub-category 2.3—Multiple input methods

Sub-category 2.3 concerns the ability of the application to provide its functionality while accepting multiple input methods.

5.2.3.1 SNAP Counting Unit (SCU)

The SCU is the EP.

5.2.3.2 Vocable

The input method is a technique or media type, which is used to deliver data into the assessed application, such as bar code reader, fax, pdf, office document, screen, voice message, SMS, or smart mobile device. The assessed application may need to identify the input method in order to interpret and use the received information.

5.2.3.3 Complexity parameters

The following are complexity parameters:

a) The number of data element types (DETs) in the SCU;

b) The number of additional input methods.

5.2.3.4 SNAP Points (SPs) Calculation

Identify the complexity based on the number of DETs. Calculate SP based on the constant factor and the number of input methods (Table 13).

Table 13 — SNAP sizing for multiple input methods

Input methods complexity	Low	Average	High
DETs	1–4 DETs	5–15 DETs	16+ DETs
SP	3 × the number of additional input methods	4 × the number of additional input methods	6 × the number of additional input methods

5.2.3.5 Rules

The multiple input methods sub-category shall be used when multiple types of inputs invoke the same functionality. If the different input types differ in terms of DETs, FTRs, and processing logic, then they would already have been accounted for as separate functions in the FP counting process.

If the inputs are the same, then multiple input methods shall be used.

Check the following:

a) Approach taken for FP counting - single instance or multiple instances. See "Considerations for Counting with Multiple Media"[23] for details.

   The multiple methods of input for the same functionality (same DETs, FTRS and processing logic) have not been included in FP size calculation. In other words, if the FP count has been done using the single instance approach for different media types, then the additional input method of the same data entry needs to be accounted for using SNAP. For example, the same input can be provided via a smartphone or a web screen.

b) If multiple input methods are already accounted for in the FP count or the multiple instance approach has been taken for FP counting, then it shall be excluded from the SNAP sizing.

   See the Example in 5.2.4.5.

5.2.4 Sub-category 2.4—Multiple output methods

Sub-category 2.4 concerns the ability of the application to provide its functionality while using multiple output methods.

5.2.4.1 SNAP Counting Unit (SCU)

The SCU is the EP.

5.2.4.2 Vocable

The output method is a technique or media type, which is used to deliver data from the assessed application, such as Fax, PDF, Office document, screen, voice message, or SMS. The assessed application may need to manipulate the sent information in order to send it to the various outputs.

5.2.4.3 Complexity parameters

Complexity parameters are as follows:

a) The number of DETs in the SCU;

b) The number of additional output methods.

5.2.4.4 SNAP Points (SPs) Calculation

Identify the complexity based on the number of DETs. Calculate size based on the constant factor and the number of output methods (Table 14).

Table 14 — SNAP sizing for multiple output methods

Output methods complexity	Low	Average	High
DETs	1–5 DETs	6–19 DETs	20+ DETs
SP	3 × the number of additional output methods	4 × the number of additional output methods	6 × the number of additional output methods

When counting a new development project, the number of output methods should include the additional ones only, assuming one of the output methods is the base method. For example, if the new development uses four output methods, the number of additional output methods is three.

5.2.4.5 Rules

This category shall be used when there are multiple types of outputs used for the same functionality. If the different output types vary in terms of DETs, FTRs, and processing logic, then they would already have been counted as separate functions in the function point counting process.

If they are the same, then multiple output methods shall be used.

Check the following.

a) Approach taken for FPA—single instance or multiple instances. See "Considerations for Counting with Multiple Media"[23] for details.

b) The multiple methods of output for the same functionality (same DETs, FTRS, and processing logic) have not been included in FP size calculation. In other words, if the FP count has been done using single instance approach for different media types, then the additional output method of the same data entry needs to be accounted for using SNAP.

For example, the same output can be provided to a smartphone or to a web screen.

If multiple output methods are already accounted for in the FP count or the multiple instance approach has been taken for FP counting, then it shall be excluded from the SNAP assessment.

EXAMPLE A banking software application supports five different processes (in FP terms EPs): Create Account, Modify Account, Make Payment, End-of-Day (EOD) Account Creation summary report, EOD Credit Debit report.

At present, the three EPs of 'Create Account,' 'Modify Account,' and 'Make Payment' take input by keying in data from the keyboard. The bank wants to enhance the software to be able to accept input for these three processes in the form of scanned documents and by reading a barcode as well.

(The "Create Account" and "Modify Account" processes 20 DETs each, and "Make Payment" process processes 15 DETs).

The "EOD Account Creation Summary Report" and "EOD Credit Debit Report" are currently sent out in printed CSV format. The bank wants to enhance the software to be able to produce the output for these processes in the form of printed PDF as well as inline mail to the recipients.

(The EOD Account Creation Summary Report has 15 DETs and EOD Credit Debit Report has 10 DETs).

The design solution for this requirement involves two subcategories: 2.3 (multiple input methods, Table 15) and 2.4 (multiple output methods, Table 16).

Table 15 — Example: SNAP sizing for multiple input methods

No.	SCU Description	Complexity Level	# Additional Input Methods	Formula	SP
1	Create Account	High	2	6 × the additional input methods	12
2	Modify Account	High	2	6 × the additional input methods	12
3	Make Payment	Average	2	4 × the additional input methods	8

Table 16 — Example: SNAP sizing for multiple output methods

No.	SCU Description	Complexity Level	# Additional Output Methods	Formula	SP
4	EOD Account Creation Summary Report	Average	2	4 × the additional input methods	8
5	EOD Credit Debit Report	Average	2	4 × the additional input methods	8

Total SNAP size for the project = ∑ SP for Sub-category 2.3 + ∑ SP for Sub-category 2.4

12 + 12 + 8 + 8 + 8 = 48

5.3 Category 3: Technical environment

The technical environment category relates to aspects of the environment where the application resides. It assesses technology as well as changes to internal data and configuration that do not provide added or changed functionality from a function points perspective.

5.3.1 Sub-category 3.1: Multiple platforms

Software operations can be provided to support the ability of the software to work on more than one platform. For the software to be considered multi-platform, it should be able to function on more than one computer architecture or operating system. This can be a time-consuming task given that different operating systems have different application programming interfaces or APIs (for example, Linux[3)] uses a different API for application software than Windows does).

5.3.1.1 SNAP Counting Unit (SCU)

The SCU is the EP.

5.3.1.2 Vocable

Computing platform includes a hardware architecture and a software framework (including application frameworks), where the combination allows software, particularly applications software, to run. Typical platforms include a computer's architecture, operating system, programming languages and related user interface (run-time system libraries or graphical user interface).

Software platform is a framework used for the software application development. Different programming languages can be grouped into several platforms based on the programming language family.

A programming language can belong to a particular software language family like the following:

Object-Oriented: Java, C++, C#, JavaScript, Python, Smalltalk, VB, VB.NET.

Procedural: C, FORTRAN, PHP, COBOL.

Declarative: SQL, XQuery, BPEL, XSLT, XML.

Hardware Platforms: A hardware platform can refer to a computer’s architecture or processor architecture. For example, the x86 and x86-64 CPUs were one of the most common computer architectures in use in general-purpose home computers.

5.3.1.3 Complexity parameters

The complexity parameters for Sub-category 3.1 include the following.

a) Nature of platforms (i.e., software, hardware).

b) Numbers of platforms to operate.

c) Family of platforms.

5.3.1.4 SNAP Points (SPs) Calculation

Identify the different software and hardware platforms involved and the number of platforms to operate. Calculate size based on the platform category row from Table 17 and the number of platforms. If more than one row is applicable, then the size is the sum of constant factors obtained from each category applicable.

3) The information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE, ISO or IEC of these products. Equivalent products may be used if they can be shown to lead to the same results.

Table 17 — SNAP sizing for multiple platforms

	2 platforms	3 platforms	4+ platforms
Category 1: Software Platforms: Same Software Family	20	30	40
Category 2: Software Platforms: Different Family	40[a]	60[a]	80[a]
Category 3: Software Platforms: Different Browsers	10	20	30
Category 4: Hardware Platforms: Real time embedded systems	TBD[b]	TBD[b]	TBD[b]
Category 5: Hardware Platforms: Non-Real time embedded systems	TBD[b]	TBD[b]	TBD[b]
Category 6: Combination of Hardware and Software: Non-Real time embedded systems	TBD[b]	TBD[b]	TBD[b]

[a] For category 2: Software platforms: Different family, count number of different families (2 families, 3 families, 4+ families).

[b] TBD: To be defined. Currently the platforms considered in the calibration of the model are only software type platforms.

EXAMPLE 1

The following are examples of platforms from Table 17.[4)]

Software Platform: .NET, Java.

Operating System Platform: MS Windows, Linux, IBM/Microsoft Operating System 2, Mac OS.

Hardware Platform: Mainframe computers, Midrange computers, RISC processors, Mobile device architecture, Mac systems.

Browsers: Edge, Firefox, Chrome.

Working on one platform is a basic requisite for the system to operate; therefore, a single platform does not generate SPs using this sub-category.

Building the software on multiple platforms generates SPs for two platforms or more according to Table 17.

Software platforms can be added or removed, not changed. For example, upgrading from Firefox 3.1.x to 3.2.a is considered as adding a platform, not changing one.

Upgrading a software platform from one version to another is counted as adding a platform only. The old platform is not counted as deleted or as changed.

When adding or removing platforms during an enhancement project, size the multiple platform sub-category for each impacted SCU at the end of the enhancement project and use that as the SNAP size for the enhancement.

For enhancement projects, count the total number of platforms after the project is complete and use that to calculate SNAP size (do not size the changes, additions or deletions of platforms during the enhancement project).

When adding a mixture of platforms from different categories, count SPs for each platforms' category.

EXAMPLE 2

Two platforms of Family 1 and 2 platforms of Family 2:

Use category 1 (similar platforms) for each software family (20 SP for the two platforms of family 1) + (20 SP for the two platforms of family 2).

Use category 2 (different platforms—SPs for 2 different platform families (= 40 SP).

EXAMPLE 3

Three platforms of Family 1 and 2 platforms of Family 2:

Use category 1 (similar platforms) for each software family (30 SP for the 3 platforms of family 1) + (20 SP for the two platforms of family 2).

4) The information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE, ISO or IEC of these products. Equivalent products may be used if they can be shown to lead to the same results.

Use category 2 (different platforms)—SPs for 2 different platform families (= 40 SP).

EXAMPLE 4

Three platforms of Family 1 and 1 platform of Family 2:

Use category 1 (similar platforms) for each applicable software family (30 SP for the 3 platforms of family 1) + (0 SP for the 1 platform of family 2).

Use category 2 (different platforms)—SPs for 2 different platform families (= 40 SP).

EXAMPLE 5[5]:

Three platforms of Family 1, 1 platform of Family 2 and 2 browsers:

Use category 1 (similar platforms) for each software family (30 SP for the 3 platforms of family 1) + (0 SP for the platform of family 2).

Use category 2 (different platforms) - SPs for 2 different platform families (= 40 SP) Use Category 3 for the browsers support (10 SP).

If an application is built on JAVA and COBOL and requires multiple (more than 4) browser support, then SNAP size would be 40 SP per each SCU that is built on both Java and Cobol, (Java and COBOL are considered as a different family plus 30 SP per each SCU that is to work on multiple browsers.

This sub-category shall be used only if the same set of functionalities is being delivered on multiple platforms. This is the case where business functionality is the same, but it needs to be delivered in two different environments. For example, if the same application functions are built on JAVA and also on VC++ to suit client requirements, then this category can be used.

If the architectural framework itself consists of different platforms to deliver part of the functionality, then this category shall not be used. This is a usual case where different technical components interact with each other to deliver application functions. No duplication of effort takes place to rebuild the same functionality in a different environment.

5.3.2 Sub-category 3.2: Database technology

The database technology sub-category comprises features and operations that are added to the database or to the statements to read/write data to and from the database to deliver NFR without affecting the functionality that is provided.

5.3.2.1 SNAP Counting Unit (SCU)

The SCU is the EP.

5.3.2.2 Database changes: each of the following sub-items is considered as one change.

a) Creating or changing a business table or a reference table, such as the following:

   1) Adding tables or adding columns for non-functional purposes only;

   2) Rearranging the order of columns in a table;

   3) Changing or adding relationships using referential integrity features;

   4) Changing the primary key without dropping and adding the primary key.

b) Creating or updating a code data table, such as the following:

   1) Adding tables or adding columns for non-functional purposes only;

5) The following information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE, ISO or IEC of these products. Equivalent products may be used if they can be shown to lead to the same results.

2) Rearranging the order of columns in a table;

3) Changing the primary key without dropping and adding the primary key.

c) Adding, deleting, or changing an index, such as:

1) Changing the columns used for indexing;

2) Changing the uniqueness specification of an index;

3) Clustering the table data by a different index;

4) Changing the order of an index (ascending or descending).

d) Adding or changing database views and partitions, such as:

1) Changing or adding database partitioning;

2) Adding, changing, or removing a database view.

3) Changing database capacity, such as:

i) Table space;

ii) Enhancing the performance features.

e) Changing a query or insert, such as changes to queries or data selection or inserts to the database without adding, deleting, or changing functionality.

EXAMPLE Changing the primary key and adding a relationship is counted as one change.

5.3.2.3 Complexity parameters

Complexity parameters for sub-category 3.2 include the following:

a) FTR complexity;

b) The number of database-related changes.

Changes to the database may be done for any non-functional requirement, such as performance, capacity management, or data integrity. The complexity of implementing any such change depends on the complexity of the Logical File as well as the number of changes (Table 18).

Table 18 — FTR complexity, database technology

DET	1-19	20-50	>50
1 RET	Low	Low	Average
2–5 RET	Low	Average	High
>5 RET	Average	High	High

5.3.2.4 SNAP Points (SPs) Calculation

Calculate size as the product of the constant factor and the number of changes (Table 19)

Table 19 — SNAP sizing for database technology

FTR Complexity Factor	Low	Average	High
SP	6 × the number of changes	9 × the number of changes	12 × the number of changes

If multiple FTRs are being impacted for the NFR, which are all impacting the same EP, then the higher complexity of the FTR shall only be considered as the Complexity Factor, not the individual FTRs separately.

In such a case, perform the following steps:

a) Identify the affected FTRs for the EP;

b) Determine the complexity of the FTRs impacted;

c) Choose the highest complexity.

Use this sub-category for new development/new requirement as well as enhancement. For new development or new requirement, separate the requirement into its functional aspects and its non-functional aspects.

EXAMPLE

An EP "Create Order" is designed for performance improvement. To achieve this, a "read-only" database view is created on "Customer" FTR having 18 DETs and 3 record element types (RETs) (FTR complexity is "Low").

In addition, an index is created on "Order Placed" FTR having 30 DET and 3 RET (FTR complexity is "Average").

The highest FTR complexity is "Average;" therefore, for the two changes: SP 9 × 2 = 18.

5.3.3 Sub-category 3.3: Batch processes

Batch jobs that are not considered functional requirements (they do not qualify as a transactional function) can be considered in SNAP. This sub-category allows for the sizing of batch processes, which are triggered within the boundary of the application, not resulting in any data crossing the boundary.

NFR for the software associated with batch jobs such as improving the job completion time, increasing the capacity of the job to process higher volumes of transactions, or performance improvement requirements may be sized using SNAP Sub-categories 3.2, 1.1, or 1.2, as applicable.

However, if an NFR for the software related to batch processing is not covered under these sub-categories, it may be considered in Sub-category 3.3.

5.3.3.1 SNAP Counting Unit (SCU)

The SCU is the user-identified batch job.

5.3.3.2 Complexity parameters

The complexity parameters for Sub-category 3.3 are as follows:

a) Number of DETs processed by the job;

b) Number of FTRs either read or maintained by the job.

5.3.3.3 SNAP Points (SPs) Calculation

When several batch jobs are automated (run always as a whole) and only the end result is user identifiable, count these batch jobs as an individual SCU.

For each user-identified job, identify the complexity level by counting the number of FTRs read or maintained. Calculate size as per Table 20.

Table 20 — SNAP sizing for batch processes

Complexity level	Low (1–3 FTR)	Average (4–9 FTR)	High (10+ FTR)
SP	4 × the number of DETs	6 × the number of DETs	10 × the number of DETs

If different processes are merged into one batch, then count the DETs and FTRs in the merged batch.

User-specified one-time data loads to logical tables can be counted using this category. These data loads should not be migration-related data loads, which are counted as conversion FP using function points.

EXAMPLE

a) Intermediate data for job validation that are in code data.

b) Scheduler data instructs to perform subsequent process steps, which are in code data.

5.4 Category 4: Architecture

The architecture category relates to the design and coding techniques utilized to build and enhance the application. It assesses the complexities of modular or component-based development.

5.4.1 Sub-category 4.1: Component-based software

Component-based software is used within the boundary of the assessed application to integrate with previously existing software or to build components in the system.

5.4.1.1 SNAP Counting Unit (SCU)

The SCU is the EP.

5.4.1.2 Vocable

a) A software component is a piece of software offering a predefined service that is able to communicate with other components via standard interfaces.

b) An individual software component is a software package, a Web service, or a module that encapsulates a set of related functions (or data). The essence of a "component" is the encapsulation of business logic or technical functionality that admits a standard interface. The software component is the element that conforms to a component model and can be independently deployed and composed without modification, according to a composition standard.

c) A component model defines specific interaction and composition standards. A component model implementation is the dedicated set of executable software elements required to support the execution of components that conform to the model.

d) Criteria for software components:

   1) Performs a specific functionality;

   2) Capable of parallel execution: multiple-use;

   3) Exchangeable: non-context-specific;

   4) Composable with other components (can be selected and assembled in various combinations to satisfy specific user requirements);

   5) Encapsulated i.e., non-investigable through its interfaces;

   6) A unit of independent deployment and versioning with well-defined interfaces and communicates via interfaces only;

   7) Has a structure and behaviour that conforms to a component model like COM, CORBA, or Java[6].

EXAMPLE Figure 4 shows simple components interacting with each other.

6) The information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE, ISO or IEC of these products. Equivalent products may be used if they can be shown to lead to the same results.

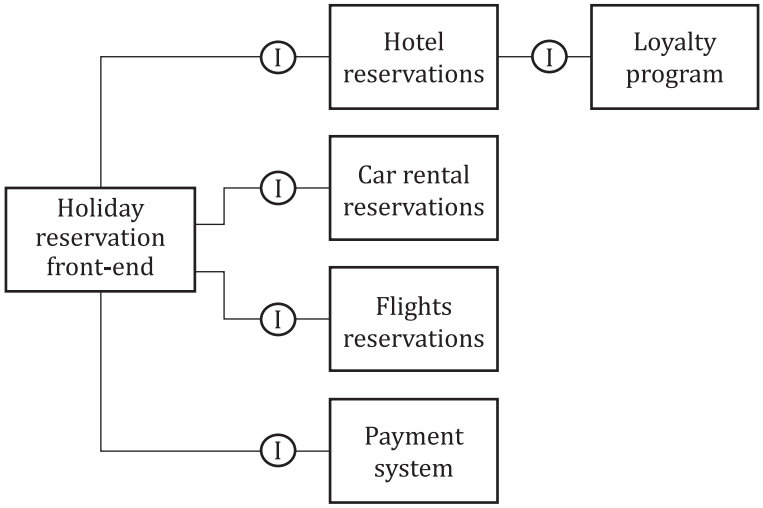


Figure 4 — Sample Components model for holiday reservation system

5.4.1.3 Complexity parameters

The complexity parameters for Sub-category 4.1 include the following:

a) Third-party component or in-house reuse;

b) The Number of unique components that are involved in the EP.

5.4.1.4 SNAP Points (SPs) calculation

Calculate size based on the constant factor and the number of unique components as per Table 21.

Table 21 — SNAP sizing for component-based software

Type	SPs calculation
In-house components	SP 3 × the number of unique components
Third-party components	SP 4 × the number of unique components

This sub-category does not size the functionality of the component. Follow the instructions of ISO/IEC 20926:2009 to count the functional size of the components.

Reuse of components may be applied to meet NFR for the software such as maintainability, changeability, maturity or replaceability.

5.4.2 Sub-category 4.2—Multiple input/output interfaces

Applications required supporting multiple inputs and outputs interfaces (user files with the same format) are covered in this subcategory. Multiple input/output interfaces can be needed due to a growing number of users and volume of data over a period of time.

Adding more input/output interfaces without changing the functionality is not considered functional change and hence such changes are not sized by FP. This sub-category shall be used to size such changes in an application.

If the project/organization considers adding new input/output interfaces as a functional change, then function points are used for sizing, and SNAP shall not be used.

5.4.2.1 SNAP Counting Unit (SCU)

The SCU is the EP.

5.4.2.2 Complexity parameters

The complexity parameters for Sub-category 4.2 include the following:

a) The number of DETs in the SCU;

b) The number of additional input and output interfaces.

5.4.2.3 SNAP Points (SPs) Calculation

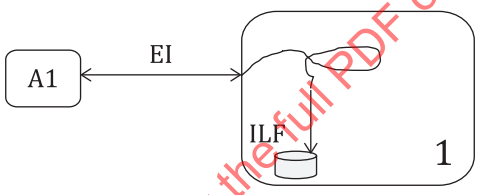
Identify the complexity based on the number of DETs in the SCU.

The size is the product of the factor derived from the number of DETs specified in Table 22 and the number of added interfaces.

Table 22 — SNAP sizing for Multiple Input /Output Interfaces

Complexity Level	Low	Average	High
DET	1–5 DETs	6–19 DETs	20+ DETs
SP	3 × the additional number of interfaces	4 × the additional number of interfaces	6 × the additional number of interfaces

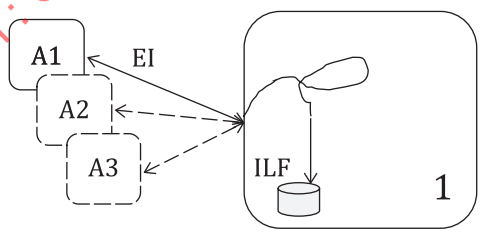
EXAMPLE 1 Adding interfaces to external application A1 without adding or changing the functionality. One SCU is shown in Figure 5; data flows to and from the boundary.



Key

1 application boundary

Figure 5 — Example 1, before the change



Key

1 application boundary

Figure 6 — Example 1, after the change

NOTE Dotted lines in Figure 6 through Figure 9 indicate the change to the existing configuration as follows:

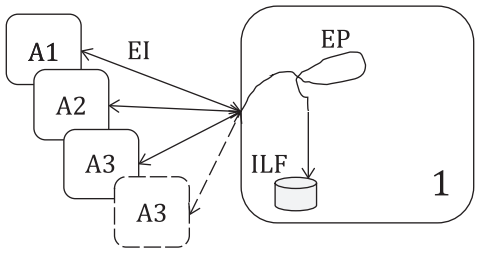
The number of DETs in the EP = 6 (average complexity)

Two additional interfaces

SP 4 × 2 interfaces = 8

EXAMPLE 2 After the change in Example 1 was delivered, it is required to add another interface to external applications A1, A2, A3.

One SCU is shown in Figure 7; data flows to and from the boundary.



Key

1 application boundary

Figure 7 — Example 2, after the second change

The change to the previous configuration is as follows:

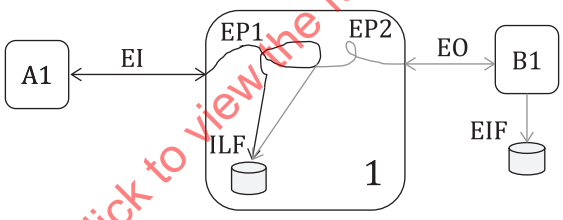
a) The number of DETs in the EP = 6 (average complexity)

b) One additional interface

SP 4 × 1 interfaces = 4

EXAMPLE 3 Adding interfaces to external applications A1 and B1 without adding or changing the functionality.

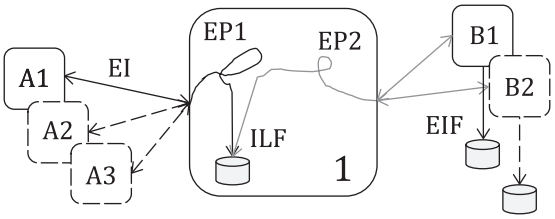
Two SCUs are shown in Figures 8 and 9; data flows to and from the boundary.



Key

1 application boundary

Figure 8 — Example 3, before the change



Key

1 application boundary

Figure 9 — Example 3, after the change

The number of DETs in EP1 flowing to and from application A1 = 5 and the number of DETs in the EP2 flowing to and from application B1 = 8.

For SCU 1 = EP1,

5 DETs = Low

2 Additional interfaces

SP3 × 2 interface = 6

For SCU 2 = EP 2,

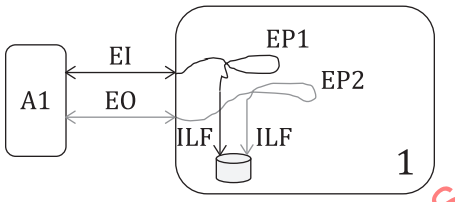
8 DETs = Average

1 Additional interface

SP 4 × 1 interface = 4

SP 6 + 4 = 10

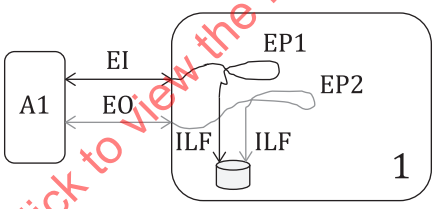
EXAMPLE 4 Adding interfaces to external application A1 without adding or changing the functionality. Two SCUs are shown in Figure 10 and Figure 11.



Key

1 application boundary

Figure 10 — Example 4, before the change



Key

1 application boundary

Figure 11 — Example 4, after the change

The number of DETs in the EI = 5 and in the EO = 10

For SCU 1 = EP1,

5 DETs = Low

1 Additional interface

SP 3 × 1 interface = 3

For SCU 2= EP2,

10 DETs = Average

1 Additional interface

SP 4 × 1 = 4

Total SP = 3 + 4 = 7

NOTE 2 A key difference between this sub-category and 2.3 / 2.4 multiple input /output methods is that in 4.2, the existing interface is replicated with the same technology to give all users the same level of performance and same experience.

5.5 Sizing code data

Code data is a type of data entities used for software sizing (in addition to business data and reference data).

According to ISO/IEC 20926:2009, code data usually exists to satisfy NFR from the user (for quality requirements, physical implementation or technical reasons).

The user does not always directly specify code data. In other cases, it is identified by the developer in response to one or more NFR.

Code data provides a list of valid values that a descriptive attribute may have. Typically, the attributes of the code data are code, description or other specified attributes describing the code (e.g., abbreviation, effective date, termination date, audit trail data). The different categories of data are outlined below to assist in identification.

When codes are used in the business data, it is necessary to have a means of translating to convert the code into something more recognizable to the user. In order to satisfy NFR, developers often create one or more tables containing the code data. Logically, the code and its related description have the same meaning. Without a description, the code may not always be clearly understood.

The key differences between code data and reference data are as follows:

a) With code data, one can substitute one for the other without changing the meaning of the business data; e.g., airport-code versus airport-name, colour-ID versus colour-description.

b) With reference data, one cannot substitute (e.g., tax code with the tax rate).

5.5.1 Code data characteristics

Code data has most of the characteristics outlined in the following subclauses.

5.5.1.1 Logical

Logical characteristics of code data include the following:

a) Data is mandatory to the functional area but optionally stored as a data file;

b) Not usually identified as part of the FUR; it is usually identified as part of the design to meet NFR;

c) Sometimes user maintainable (usually by a user support person);

d) Stores data to standardize and facilitate business activities and business transactions;

e) Essentially static—only changes in response to changes in the way that the business operates;

f) Business transactions access code data to improve ease of data entry, improve data consistency, or check data integrity.

If recognized by the user, code data is sometimes considered as a group of the same type of data, and can be maintained using the same processing logic.

5.5.1.2 Physical

Physical characteristics of code data include the following:

a) Consists of key field and usually one or two attributes only;

b) Typically has a stable number of records;

c) Can represent 50 % of all entities in third normal form;

d) Sometimes de-normalized and placed in one physical table with other code data;

e) May be implemented in different ways (e.g., via a separate application, data dictionary, or hard-coded within the software).

EXAMPLES State, State code, State name, Payment type, Payment type code, Payment description.

5.5.2 Handling code data from non-functional sizing perspective

For the purpose of FPA, code data cannot be counted as logical files. They are not considered as Internal Logical File (ILF) or External Interface File (EIF), and cannot be considered Record Element Types (RETs) or DETs on an ILF or EIF. Code data cannot be considered a File Type Referenced (FTR) while assessing the complexity of a transactional function (EI, EO, and External Inquiry—EQ).

For the purpose of SNAP, code data that is maintained within the application boundary by use of screens or by formal enhancement requests by the customer is counted as follows:

Irrespective of the number of code data physical tables, the code data shall be grouped as 1 data group (1 FTR) under SNAP

Code data is classified as in Table 23.

Table 23 — Types of code data

Substitution	Static or constant	Valid values
Code + Description	One occurrence, Static Data, Default value	Valid Values, Range of valid values

For SNAP analysis of the complexity of code data group, the number of RETs of the code table depends upon the type of occurrences of code data types.

EXAMPLE In a banking application, the following code tables were created:

Table 1: state name and state code.

Table 2: branch code, branch name, branch city.

Table 3: one single data entry of the bank name and logo, which is used for printing letterheads Three types of occurrences exist for code data:

Substitution—Table 1: state code and state name; Table 2: branch code and branch name.

Valid values—Table 2: a range of bank branch cities.

Static data—Table 3: one single data entry of the bank name and logo, which is used for printing letterheads.

Hence the number of RETs for the code data group is 3.

If the data is not of the above code data sub types, then it may be the data in system tables and not the application tables is required for supporting business. This is not sized under code data.

5.5.3 How code data is sized using SNAP

Count the creation/maintenance and the utilization of code data.

The creation of code data is always counted as Sub-category3.2: Database technology.

The maintenance of code data is checked under the following subcategories depending on the case applicable:

a) If the code data is maintained in hard-coded tables that are not viewable via screens, but can be updated by the administrator using script/code change in response to a formal user request, then it is counted using Sub-category 3.2: Database Technology.

   EXAMPLE A code table has the bank name and logo stored as static data, which is referred to by different processes. When a change request is raised to modify the logo, then it is sized using this category.

b) When the code data is used for reasons such as entry validations, data formatting, batch process management, any changes to code data values (add /change/ delete) shall be counted using the proper sub-category.

   The utilization of code data is counted in the following sub-categories, according to the purpose of the data: 1.1 Data Entry Validation; 1.2 Logical and Mathematical Operations; 1.3 Data Formatting; 1.5 Delivering Added Value to Users by Data Configuration; and 3.3 Batch Processes.

   When NFR use code data and transactions cross partitions, Sub-category 1.4 Internal Data Movements shall be used.

   Examples of SNAP sizing of code data are shown in Table 24.

Table 24 — Example of SNAP sizing of code data

Examples:	Sub-categories for utilizing code data
Create a code data table for address validation	1.1 Data Entry Validation for enabling the validation, 3.2 Database Technology for code table creation
Same as above. The screens with the address are on the Front-End application; the data is in the Back-End application	1.1 Data Entry Validation for enabling the validation, 3.2 Database Technology for code table creation, 1.4 Internal Data Movements
Using multi-language screens, the translation is in new code data tables	1.3 Data Formatting, 3.2 Database Technology for code table creation
Add scheduling data to perform batch files	3.3 Batch Processes, 3.2 Database Technology for code table creation

6 The sizing process

6.1 Introduction

The non-functional sizing process uses the collection of information, which is needed for determining what is measured and then measuring its non-functional size.

The sizing process contains the following steps:

Gather available documentation.

a) Determine the purpose, scope, boundary, and partition of the measurement.

b) Identify the NFR.

c) Associate NFR with sub-categories and identify the SCU.

d) Determine the SNAP size for each sub-category.

e) Calculate non-functional size.

f) Document and report.

6.2 The timing of the non-functional sizing

Non-functional sizing can be assessed at any time in the development lifecycle to aid in project estimating, monitoring project change of scope, and evaluating delivered NFR.

Before beginning a non-functional sizing, users should determine whether they approximate or measure the size, and document any assumptions.

Approximating permits assumptions to be made about unknown non-functional aspects, to determine an approximate non-functional size.

At an early stage, NFR may not be fully defined. Despite the disadvantages, this sizing can be useful to produce an early estimate. Uses of the non-functional sizing for approximating or measuring non-functional size at the various life cycle phases are presented in Table 25.

Table 25 — The timing of approximating or measuring the non-functional size

Life cycle phase	SPs can be approximated	SPs can be measured
Proposal: users express needs and intentions	Yes	No
Requirements: developers and users review and agree upon the expression of user needs and intentions	Yes	No
Design: developers may include elements for implementation	Yes	Yes
Construction	Yes	Yes
Delivery	Yes	Yes
Maintenance (adaptive, perfective, or preventive)	Yes	Yes

NOTE No specific development life cycle is implied. The lifecycle processes, activities, or milestones above can be used in any software development lifecycle.

6.3 Non-functional sizing and FSM

Non-functional size can be used in conjunction with functional size to provide an overall view of the project or application, including both functional and non-functional sizing.

Figure 12 illustrates the joint process.

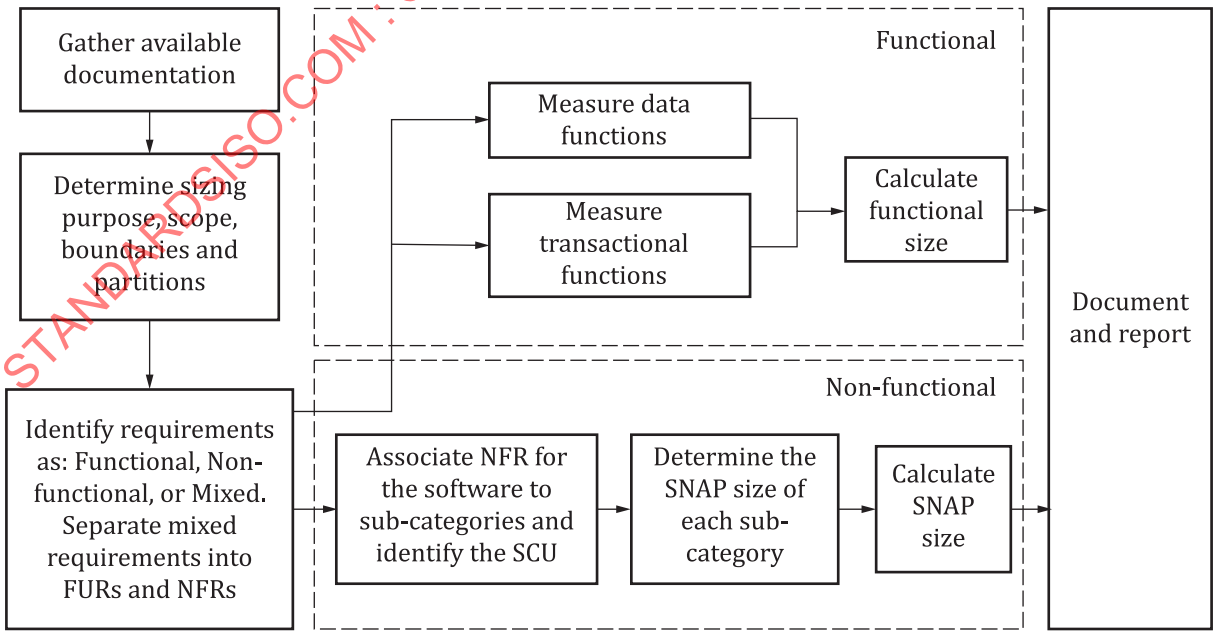


Figure 12 — The sizing process

The steps listed in 6.3 describe the non-functional sizing process.

6.4 Steps to determine the non-functional size

6.4.1 Step 1: Gather available documentation

Documentation to support the measurement process describes the NFR delivered by the software or the non-functional aspects that are impacted by the software project that is being measured. Non-functional characteristics may also be embedded in the functional documentation.

Sufficient documentation, or access to subject matter experts, who can provide additional information to address gaps in the documentation, should be obtained. Standards and International guidelines (such as WCAG 2.1 accessibility guidelines) should be considered if applicable.

NFR may be implicit or explicit. When not explicitly specified, verify that expectations are understood and documented.

NFR are subject to configuration management.

6.4.2 Step 2: Determine the sizing purpose, type, scope, boundary, and partition

6.4.2.1 through 6.4.2.4 contain required activities to determine the non-functional size.

6.4.2.1 Identify the purpose

Sizing the non-functional size may be conducted to perform the following:

a) Provide a more accurate estimation of effort and time to deliver a software project;

b) Assess the value of the installed base of applications to determine the support costs;

c) Assess the developed size and determine the cost of the project based on size,

Example "The sizing purpose is to estimate the effort needed to meet the non-functional requirements".

6.4.2.2 Identify the sizing type

The functional size and the non-functional size can be measured for either projects or applications. The type of sizing is determined, based on the purpose, as one of the following:

a) Development project sizing.

b) Enhancement project sizing.

c) Application assessment.

Figure 13 illustrates the types of sizing and their relationships. (Project A is completed first, followed by Project B.)

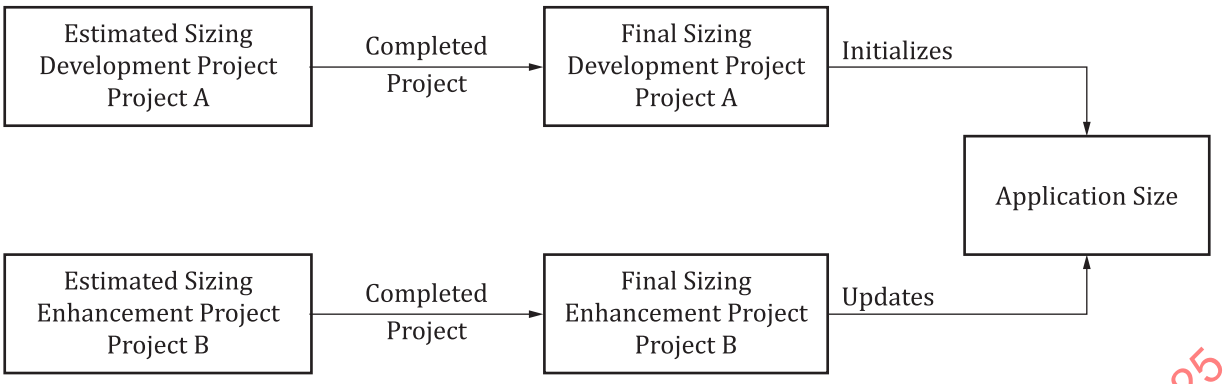


Figure 13 — Types of sizing

6.4.2.3 Identify the scope

The scope defines the set of NFR to be included in the sizing.

a) The scope:

   1) Is determined by the purpose of performing the non-functional sizing.

   2) Defines a set of partition(s).

   3) Identifies which SNAP categories and sub-categories should be included in the non-functional size measurement to measure the size for the development and delivery of the software product.

   4) May include more than one application.

b) Types of sizing include the following:

   1) Sizing a development project includes all NFR for the development and delivery of the software product.

   2) Sizing of an installed base of applications includes all NFR for the support of the installed applications.

   3) Sizing an enhancement project includes all NFR for the development and delivery of the enhancement project.

   4) Sizing a maintenance project includes all NFR for a selected scope.

To help identify the processes sizing scope:

Review the purpose of the non-functional sizing to help determine the scope.

When identifying the scope for measuring the non-functional size of the installed base of applications, include all of the non-functional categories supported by the maintenance team, eventually distinguished by partition within each application's boundary.

6.4.2.4 Determine the boundary

Determine the boundary of each application within the sizing scope, based on the user view.

a) In order to establish the boundary, the user view should be defined. The following hints can help to identify the user view:

   1) A user is any person or thing (application, device) that communicates or interacts with the software at any time.

   2) A user view consists of the functional and NFR as perceived by the user.

3) A user view represents a formal description of the user's needs in the user's language.

4) A user view can be verbal statements made by the user as to what their view is.

5) A user view should be approved by the user.

6) A user view can vary in physical form (e.g. user stories, catalogue of transactions, proposals, requirements document, external specifications, detailed specifications, user handbook, quality or non-functional specifications).

7) A partition may act as an "internal user" for another partition within the same application boundary, in terms of data exchange or data sharing; consequently, different non-functional sizing may be created for each partition.

b) Establish the boundary. The boundary (also referred to as application boundary):

1) Defines what is external to the application.

2) Acts as a "membrane" through which data processed by transactions pass into and out of the application

3) Is dependent on the user's external business view of the application; it is independent of non-functional or implementation considerations

The positioning of the boundary between the software under investigation and other software applications may be subjective. It is often difficult to delineate where one application stops and another begins. Try to place the boundary from a business perspective rather than based on technical or physical considerations.

EXAMPLE Figure 14 shows the boundaries between the Human Resources application and the external applications, Currency and Fixed Assets. The example also shows the boundary between the human user (User 1) and the Human Resources application. The Human Resource application can in turn internally satisfy the functional, technical and quality requirements specified by the user.

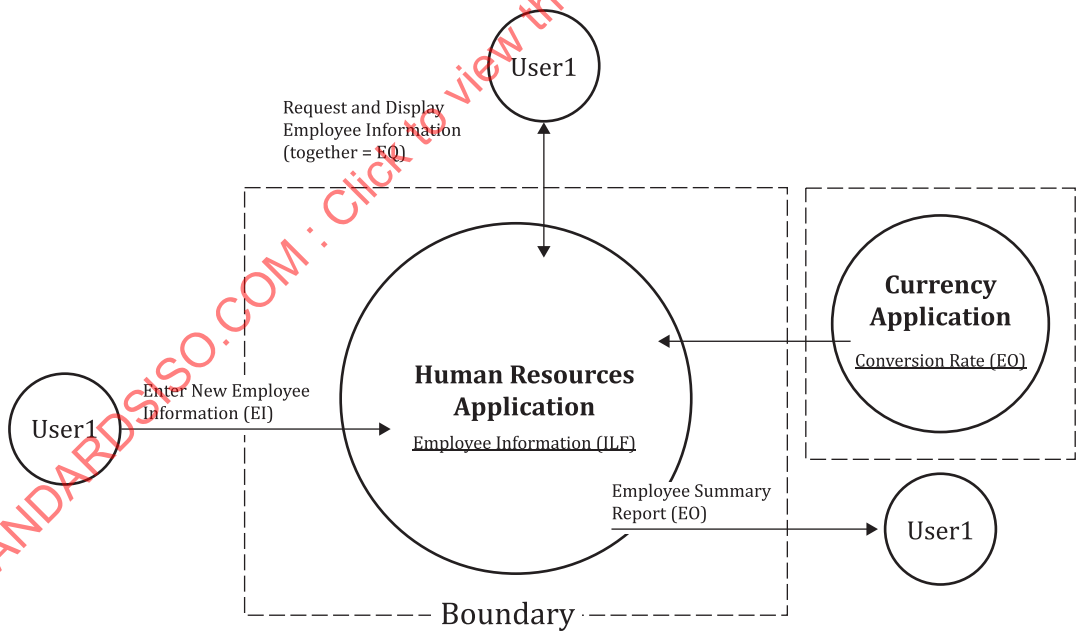


Figure 14 — Boundary example

c) Apply rules for boundaries. The following rules shall apply to boundaries:

1) The application boundaries shall be consistent between the FPA and SNAP processes.

2) The boundary shall be determined based on the user's view; the focus is on what the human user can understand and describe.

3) The initial boundary already established for the application, or applications being modified, shall not be influenced by the sizing scope.

4) There may be more than one application included in the sizing scope. If so, multiple application boundaries shall be identified.

5) When the boundary is not well-defined (such as early in the analysis), it shall be located as accurately as possible.

d) Using a multiple-instance approach versus a single-instance approach

Different organizations may take different approaches for sizing similar functionality being delivered on different media. They use the single instance or the multiple instance approaches to specify the same. Single instance approach is said to be when the same functionality is delivered via different mediums (input or output), but is counted only once. Organizations using the single instance approach for the FP size can size the other methods of delivery using SNAP. The single instance approach does not recognize the medium for delivery for a transaction function as a differentiating characteristic in the identification of unique transaction functions. If two functions deliver the same functionality using different media, they are considered to be the same function for functional sizing purposes. The multiple instance approach specifies that instance functional size is taken in the context of the approach objective of the count, allowing a business function to be recognized in the context of the medium in which it is required to operate. The multiple instance approach recognizes the medium for delivery for a transaction function as a differentiating characteristic in the identification of unique transaction functions.

6.4.2.5 Identify the partitions, if applicable

When identified, partitions may add non-functional size. Sub-category 1.4 (Internal Data Movements) is used to provide the additional non-functional size for the application being assessed.

The following hints can help to identify the boundary and the partition of the application(s):

a) Use the system external specifications or a system flow chart and draw a boundary around it to highlight which parts are internal and which are external to the application.

b) Look at how groups of data and software partitions are being maintained.

c) Identify functional areas by assigning ownership of certain types of analysis objects (such as entities or EPs) to a functional area; non-functional categories are determined by the identification of the functional boundaries (application boundaries as determined by FPA) and eventually partitions within them.

d) Look at the associated measurement data, such as effort, cost, and defects; the boundaries for measurement should be the same, or eventually, those measurement data may be distinguished by partitions within a single boundary.

e) Interview subject matter experts for assistance in identifying the boundary.

f) Interview software analysts for assistance in identifying the partitions, if any.

Partitions shall not overlap.

6.4.3 Step 3: Identify the NFR

The following guidelines may be used to identify all NFR:

a) NFR may be explicitly documented but may also be implicit. Users should make sure that they identify all requirements, implicit or explicit.

b) NFR may be mandated by law or by customer's policy and can be undocumented as part of the project's documentation.

c) NFR and functional requirements should be separated. However, in actual project documentation, a requirement may contain both functional and non-functional aspects.

d) A requirement should be broken down into its functional components and non-functional components, and the segregation should be agreed by both the user/customer and development teams.

NOTE Rules for segregating the functional aspect of the requirements from its NFR are provided in 7.1.

6.4.4 Step 4: Associate NFR with sub-categories and identify the SCU

Identify the subcategories as follows:

a) Identify the applicable sub-categories for the NFR. Use the description of the sub-categories and the examples in this document to associate NFR with sub-categories.

b) Identify the Counting Units (SCUs).

The SCUs are unique to each sub-category; they are determined by the nature of the sub-category. The SCU is part of the sub-category definition.

A non-functional requirement may be met using several SCUs. For example, a requirement for improved time behaviour (performance) may involve several EPs, each EP is an SCU.

Sizing is done separately per each SCU.

6.4.5 Step 5: Determine the SNAP size for each sub-category

For each SCU:

a) Identify the complexity parameters of the SCU.

b) Use Clause 5 to determine the non-functional size of the SCU.

The non-functional size for the sub-category is the sum of the SPs of all SCUs.

6.4.6 Step 6: Calculate the non-functional size

The non-functional size is the sum of the sizes of all SCUs identified in all sub-categories (see 6.4).

6.4.7 Step 7: Document and report

The sizing results shall be documented as follows:

a) The purpose and the sizing type;

b) The measurement scope and the boundary of the application;

c) The date of the measurement;

d) A list of all SCUs, including their type and complexity and number of SPs assigned;

e) The result of the measurement;

f) Any assumptions made and issues resolved.

The documentation may also include identification of the following:

a) Source documentation on which the count was based;

b) Participants, their roles and qualifications.

Negotiate the level of documentation with the client, and inform the client about the related costs and benefits.

A fully documented measurement should facilitate traceability, usability and maintainability; however, a client may be interested only in the results.

The document should have versioning control.

The document should state the conventions to be adopted when reporting size such that it is qualified with:

a) The units of the size

b) The name of the sizing method

EXAMPLE Non-functional size = 300 SPs (ISO/IEC/IEEE 32430:2024).

6.5 Calculating the non-functional size

6.5.1 Formula approach

Determine the non-functional size in four steps:

1) Step 1. For each requirement, identify the categories and sub-categories that are associated with the requirement.

2) Step 2: For each of the sub-categories, identify the SCUs.

3) Step 3: Determine the non-functional size (SPs) for each SCU within the sub-category, by using the equation or the table for the sub-categories.

4) Step 4: Determine the SPs for a specific project or application by using the formula for the project type in question.

6.5.2 Determine the non-functional size of each sub-category

The non-functional size of each sub-category shall be determined using the defined measure for the SCU for each sub-category.

There is one definition of the SCU for each of the sub-categories, as defined in the sub-category definition.

The size of each sub-category is determined by using the defined equation or table for each sub-category.

6.5.3 Determine the non-functional size of a development project

The size of the NFR is equal to the sum of SP sizes of each category. A development project non-functional size shall be calculated using the development formula.

The formula for development project:

$$S_{DSP} = S_{ADD}$$

where

S_{DSP} is the development project SNAP size

S_{ADD} is the size of the NFR delivered to the user by the development project

S_{ADD} is the sum of SP for all sub-categories

The application non-functional size is equivalent to the non-functional size of the development project.

For non-functional size, converted functionality is not identified.

6.5.4 Determine the non-functional size of an enhancement project

Enhancement projects can involve additions, changes to, and deletion of existing non-functional features.

An enhancement project is a project to develop and deliver maintenance. It may be adaptive, preventive or perfective maintenance.

The enhancement project non-functional size is a measure of the non-functional characteristics added or deleted at the completion of an enhancement project as well as changes made to existing non-functional characteristics as measured by the enhancement project SNAP size.

Enhancement NFR shall be measured in accordance with the following:

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

b) Measure requirements that are added, changed or deleted.

The application non-functional size shall be updated to reflect the following:

a) Added NFR, which increase the application non-functional size.

b) Changed NFR, which may increase, decrease or have no effect on the application non-functional size.

c) Deleted NFR, which decrease the application non-functional size.

This rule shall be applied per each sub-category.

An enhancement project non-functional size shall be calculated using the following formula:

$$S_{ESP} = S_{ADD} + S_{CHG} + S_{DEL}$$

where

S_{ESP} is the enhancement project SNAP size.

S_{ADD} is the non-functional size being added by the enhancement project.

S_{CHG} is the size of the changes made to existing NFR by the enhancement project.

S_{DEL} is the non-functional size deleted by the enhancement project.

For a sub-category SC:

$$S_{ESPSC} = S_{ADDSC} + S_{CHGSC} + S_{DELSC}$$

For the enhancement project:

$$S_{ASP} = \sum S_{ESPSC1.1} + \sum S_{ESPSC1.2} + \cdots\cdots + \sum S_{ESPSC4.2}$$

An application non-functional size after an enhancement project shall be calculated using the formula:

$$S_{ASPA} = S_{ASPB} + (S_{ADD} + S_{CHGA}) - (S_{CHGB} + S_{DEL})$$

where:

S_{ASP} is the project SNAP size.

S_{ASPA} is the application SP after the enhancement project.

S_{ASPB} is the application SP before the enhancement project.

S_{CHGA} is the size of the NFR being changed by the enhancement project as they are or will be after implementation.

S_{CHGB} is the size of the NFR being changed by the enhancement project as they are/were before the project commenced.

S_{DEL} is the size of the NFR deleted by the enhancement project.

The size of changes made to sub-categories in an enhancement project (i.e., CHG) does not provide an assessment of the overall size of the sub-categories before (CHGB) or after (CHGA) the enhancement project. CHG measures the size of the change in the non-functional characteristics being changed.

A demonstration of a situation in which the size of the change in the project (CHG) is 12 SP, but the overall size of the sub-category is 0 (CHGA = CHGB) is shown in EXAMPLE 2.

Therefore, in order to maintain the application's non-functional size after each enhancement project, one should measure, in addition to the size of the changes made, the size of the sub-category before and after the enhancement.

EXAMPLE 1 An application has an EP in which there are 10 DETs. Three DETs are already encrypted (encryption complexity is average).

The enhancement project requires to change the encryption type (new encryption is local).

a) Counting the enhancement project:

Type of requirements: CHG.

Transformation complexity: High.

$S_{ESP} = S_{ADD} + S_{CHG} + S_{DEL}$

$= 0 + 5 \times n_{DET} + 0 = 15$ (n_{DET} is the number of DETs)

b) Counting the application:

$S_{ASPB} = 3 \times n_{DET} = 9$ (Assuming this is the only NFT in the application).

$S_{CHGB} = 3 \times n_{DET} = 9$ $S_{CHGA} = 15$.

$S_{ASPA} = S_{ASPB} + (S_{ADD} + S_{CHGA}) - (S_{CHGB} + S_{DEL})$.

$= 9 + (0 + 15) - (9 + 0) = 15$.

EXAMPLE 2 A "Search" application has an entry screen with one entry: a string to search. To keep the look-and- feel of the application fresh, each month the background colour is changed, the size and location of the search field are changed and the shape and location of the search button are changed.

To build this screen, the sum of the number of unique properties is 20.

a) Counting the enhancement project: Type of requirements: CHG

UI Type complexity: High.

Number of unique UI elements: 3 (Screen, field, control);

$S_{ESP} = S_{ADD} + S_{CHG} + S_{DEL}$

$= 0 + 4 \times 3 + 0 = 12$.

b) Counting the application:

$S_{ASPB} = 3 \times 4 = 12$ (Assuming this is the only NFT in the application) $S_{CHGB} = 3 \times 4 = 12$.

$S_{CHGA} = 3 \times 4 = 12$.

$S_{ASPA} = S_{ASPB} + (S_{ADD} + S_{CHGA}) - (S_{CHGB} + S_{DEL})$.

$= 12 + (0 + 12) - (12 + 0) = 12$.

(No change in application size.)

7 Complementarity of the functional and the non-functional sizes

7.1 General

Documented requirements, in actual projects, may combine functional and non-functional aspects. Such a requirement shall be broken down into its functional components and non-functional components, and the segregation should be agreed by both the user/customer and development teams. Use FP for the functional parts of the requirements and SPs for the non-functional parts of the requirements.

Table 24 is a guideline. To define NFR, ISO/IEC/IEEE 29148 or a similar standard may be used.

Table 24 — IFPUG FPA and SNAP interrelations

Case #	Circumstance	Description	Guideline
1	Requirements are functional only	The users do not have any explicit or implicit NFR	Count function points only
2	Requirements are clearly marked as NFR	Parties agree on clear segregation between functional requirements and NFR. Requirements classified as NFR cannot be sized with function points	Count SPs only
3	Requirements involve both functional and non-functional aspects	Functional requirements have additional NFR which can be clearly identified.	See 7.2
4	Requirements are functional only, transactions cross partitions	Functional requirements may involve single or multiple flows. In case of multiple flows, and using ISO/IEC 20926:2009 (IFPUG FSM) guidelines, each flow does not qualify as a separate EP.	Count function points to size the new/enhanced functionality for the main EP as per ISO/IEC 20926:2009, add SNAP size for the transactions/ flows within the application's boundary, that cross the partitions
5	Requirements are functional, but they are provided without any software change	Functionality (or any business value) that is added or modified by changing reference data or other means that cannot be sized by FPs, according to ISO/IEC 20926:2009 guidelines or FP counting practices of the organization.	Count SPs using Sub-category 1.5—Delivering Functionality by Data Configuration

If an organization has developed their own organization guidelines based on ISO/IEC 20926:2009 (IFPUG FSM), and some non-functional aspects have been included in the FP counting, the organization should revise the guidelines to count both FPs and SPs based on ISO/IEC 20926:2009 and this document.

The following subclauses contain guidelines that should be used to determine how FP and SPs should be counted.

NOTE Having the same person count both FPs and SPs for a requirement can help to avoid double counting and provide a more efficient and accurate count of each type.

7.2 Requirements involving functional and non-functional requirements

7.2.1 Sub-category 1.1 data entry validation

7.2.1.1 What to check

Check the EP.

7.2.1.2 Rules

a) Count SPs when:

   1) Adding a new DET with validation (count SPs in addition to FPs).

   2) Changing a DET (e.g., Masking, length, format) and changing the entry validation.

   3) Changing the validation logic of an existing DET.

b) For an EP with a DET added to current functionality, count FPs per ISO/IEC 20926:2009 (IFPUG FSM) for the functional aspect of adding the DET and SPs for the complexity change of the validation.

c) When adding or changing a validation on an existing DET, count FPs per ISO/IEC 20926:2009 for the functional aspect of changing the validation logic and SPs for the impacts to change in validation complexity.

Data entry validation is counted in this subcategory only if it involves adding, changing or removing validation logic related to what data is being entered.

Adding a rule to validate that a numeric entry is not negative, in addition to validating that the entry is numeric and is decimal, is SP-counted. Adding a valid value to an existing list without changing the validation logic does not generate SPs in this sub-category (adding a valid value is counted under Sub- category 3.2 Database Technology, as a database change). Adding a new list of valid values for an existing DET to validate values against is counted in SPs both in Sub-category 1.1 and 3.2.

EXAMPLE 1 A change in a date field in an enhancement project: former validation for date entry checked the format of DD/MMM/YY and the new format should be DD/MMM/YYYY. For the EP that enters the date, this change is counted in SPs using Sub-category 1.1 in addition to FPs counting. (Count FPs if there is a change in the processing logic, as per ISO/IEC 20926:2009.)

EXAMPLE 2 In an enhancement project of a travel application for international flight booking function, passport type field is being added. A new code table is added to store passport type, e.g, regular or diplomatic. Based on passport type, validation rules exist to check the format of passport number being entered is correct or not. Two EPs are impacted: “Enter traveller's details” and “modify traveller's details”. For both EPs, count FPs for adding the new DET, SPs (Sub-category 1.1) for the data entry validation required for passport type field, and SPs (Sub-category 3.2) for adding the code table.

EXAMPLE 3 In a development project of a telecommunication self-service application, payment screen (one EP, “enter payment details”): validate that the mobile number is numerical, and is structured as “0XX- XXXXXXX” (the first digit must be zero, following 9 digits only) is SPs-counted using this sub- category in addition to FPs counting.

EXAMPLE 4 In both development and enhancement projects of a health insurance application, entering a new claim process has many fields. Validating that all fields are populated with the right format and that all mandatory fields are not empty is SP counted using this sub-category in addition to FPs counting.

EXAMPLE 5 A change is requested to the values in a drop-down list with five values, change one value and add two new values: if data entry validation logic does not change, use Sub-category 3.2 and not this sub- category to count SPs.

EXAMPLE 6 An existing EI was enhanced by adding a validation to an existing field to confirm that the value is between 8 and 12 digits long. This new validation was the only change made to the EI. This change was sized using FPs since there is a change in the processing logic as per ISO/IEC 20926:2009 (IFPUG FSM), and it was sized using SPs since it involves a non-functional aspect as well.

7.2.2 Sub-category 1.2 logical and mathematical operations

7.2.2.1 What to check

Check the EP.

7.2.2.2 Rules

Either for a development of a new request or for an enhancement, count the functional requirement using FPs for any EO/EI/EQ per ISO/IEC 20926:2009 (IFPUG FSM); in addition, count SPs when processing logic includes extensive mathematical or extensive logical operations as defined in 5.1.2. The complexity of the algorithmic calculations or the logical processing required is the non-functional aspect of the requirement.

EXAMPLE A project management software application currently allows the user to compute project completion time using the algorithm "Critical Path Method." The user now wants an enhancement to allow the new, optional use of the "Program Evaluation Review Technique" (PERT) algorithm to compute project completion times. New DETs are added to capture the attributes needed for calculating project completion time and the probability to meet the completion time.

In this case, count both FPs and SPs for each EP that is affected by the enhancement. Function points are impacted for processing logic change as there is a change in the calculation as per definition in ISO/IEC 20926:2009. SPs need be counted to assess the complexity change in the algorithmic processing being done.

7.2.3 Sub-category 1.3 data formatting

7.2.3.1 What to check

Check the EP.

7.2.3.2 Rules

a) When formatting is requested, break the requirements into its functional part (The "what") and its non-functional part (the formatting, such as encryption /decryption, byte padding, adding header/ footer information on a transaction).

b) Enhancements may have a functional aspect only (no requirements for any format) or both functional and non-functional aspects; enhancement may have a functional aspect only (such as a change in the data transferred) or non-functional aspect only, or both.

EXAMPLE 1 Billing report: To improve the look and feel of the report, the user requests an enhancement: all the fields should apply bytes padding, to help ensure the following:

a) a 20-character display field length for name field.

b) a 15-character display field length for the bill values (amounts) and the PID[7)].

c) The patient's name should be displayed as First Name. Middle Name. Last name instead of FirstName_LastName.

In this case, the requirement should be broken into two parts:

a) Functional: the patient's name should be displayed as First Name.Middle Name.Last name.

b) Non-functional: byte padding is added to help ensure a 20-character display field length for name field and 15-character display field length for amount and PID.

Count FPs to size the transaction for part a): Display Report; count SPs to size part b), using Sub- category 1.3.

EXAMPLE 2 A new HR system is being built to replace two outdated applications. To protect sensitive information, the user requests that the employee's identification number be masked on all display screens. When updating employee information, the user is provided with an option to unmask the identification number.

In this case, the requirement should be broken into two parts:

Functional: display screens and Employee Update screens should be counted under FPs.

7) PID – Patient Identifier.

Non-functional: The masking[8] and unmasking of the identification number are considered non-functional but should only be counted one time under SNAP.

EXAMPLE 3 The user has requested changes to the existing display screens that contain employee data and the "Employee Update" screen. In order to protect sensitive information, the user requests that the employee's identification number be masked on all display screens. When updating employee information, the user is provided with an option to unmask the identification number.

In this case, there is only a non-functional requirement for the masking and unmasking of the identification number. Since there is no addition/deletion of DETs or change in the processing logic for the screens, there is no change to the system's user functionality, therefore no FPs are counted.

Impacted EPs: "Add new employee," "View employee personal details," and "Edit employee personal details." For each EP:

Functional: none.

Non-functional: the masking and unmasking of the social security number are considered non-functional and should only be counted using SNAP.

7.2.4 Sub-category 1.4 internal data movements

7.2.4.1 What to check

Check the EP.

7.2.4.2 Rules

Count SPs when partitions are defined and the EP crosses partitions. Use this sub-category whether a new process is added that cross partitions, or when an existing process is either enhanced or deleted, and it crosses partitions. In addition, count FPs if the processing logic of the EP was modified as per ISO/IEC 20926:2009). The FP counting measures the data movements in and out of the boundary, and the SP counting measures the internal data movements between partitions.

Count SPs when a data flow between partitions is added, changed, or deleted to meet the non-functional requirement. (See Example 2 in 7.4.3.)

EXAMPLE 1 An HR application consists of one application boundary with the front end and the back end as two partitions. A new data entry field is added and is used by three EPs: Add, change, and enquire employee. All the employee-related transactions receive input from the front-end application and processed with data from the backend. Since there is a new field added, count FPs for the functional change in the three EPs. In addition, count SPs for the change in internal data movements in these three EPs.

EXAMPLE 2 An HR application consists of one application boundary with the front end and the back end as two partitions. A screen displays a value D when D is calculated as A/(B+C). A, B and C are retrieved from the back-end to the front-end; the calculation is performed by the front-end. To improve performance, it is requested that the value D is calculated at the back-end; data flow to the front end is changed from (A, B and C) to (D).

This screen is used by three EPs.

Count SPs for the data flow change using Sub-category 1.4 per each EP.

7.2.5 Sub-category 1.5 delivering added value to users by data configuration

7.2.5.1 What to check

Check the EP.

8) Data masking or data obfuscation is the process of hiding original data with random characters or data. The main reason for applying masking to a data field is to protect data that is classified as personal identifiable data, personal sensitive data or commercially sensitive data; however, the data must remain usable.

7.2.5.2 Rules

A software application may be designed in a way that user value can be added or changed by adding or changing data in reference tables, without any change to the processing logic. According to ISO/IEC 20926:2009, FPs cannot be counted in this case. Count SPs in such cases.

When functionality is added or enhanced using application source code change along with application configuration or addition of data to reference tables, and it generates FPs, do not add SNAP size.

If the user value added requires application source code change and the user value added using configuration are independent of each other, then count the functional change developed by changing the source code with FPs, and the change added by configuration using SPs.

EXAMPLE 1 CommWorld Company is a telecommunications company. Its BSS application enables CommWorld employees to manage products by geographical location. A new location for product setup can be enabled by adding data to the "Products Profile" table. This table stores data on which product setup functions are available for each location. The company has now expanded into the Asia Pacific Area; for the India location it must enable its product setup web pages for DSL packages; for Singapore, it must enable its product setup web pages for DSL and Cable networks.

The application maintenance team creates one entry in the "products profile" table for the India location and two entries for the Singapore location corresponding to each product type. Since this change is enabled only by adding values to the table, FPs cannot be counted. SNAP is used for sizing this change.

EXAMPLE 2 For adding or changing the product setup, the system is required to check internally that if same product package is being created in new geography as "already for sale in another location," then the corresponding row in the "product package popularity" master table is updated.

For the second part of the requirement, the new validation and update to product popularity master is a functional change to "add new product package" transaction. Hence, it is FP countable. Since the reference table updates are accompanied by application source code change and they are not independent of each other, SPs cannot be counted here, count only FPs for the entire requirement.

7.2.6 Sub-category 2.1 user interfaces

7.2.6.1 What to check

Check the set of screens as defined by the EP.

7.2.6.2 Rules

a) Creation of a new user interface (UI) element in order to add, create or change functionality, which is countable using FPs: count FPs and SPs (FPs for the functional requirement and SPs for configuring the UI element to meet NFR).

b) Creation of a new UI element that does not add or change functionality (such as adding a static menu): count SPs only.

c) In case of modification of a UI element:

   1) If functionality is added or changed and properties of the UI element are changed (see definition of properties in 5.2.1), separate the requirements into its functional aspects (counted using FPs) and its non-functional aspects (counted using SPs). SPs would assess non-functional impacts of the change in UI elements.

   2) If functionality is not changed and only properties of the UI element are changed (see definition in 5.2.1) then count the change using SPs.

EXAMPLE 1 ABC Company merged with StarY Company. UI standards followed by the application development team have changed for font size, logo and background colour. The application software must be modified to meet the latest version of the UI standards. Identify all affected EP. Per each EP, count SPs only to size the UI enhancement requirement.

EXAMPLE 2 ABC Company merged with StarY Company. UI standards followed by the application development team have changed. A new field “Customer Loyalty Points” is added to the “View Bill” transaction. FPs are counted for the change (EO) to “View Bill” transaction for the newly added field. SPs are counted for the UI element impacts for the new field “Customer Loyalty Points.”

EXAMPLE 3 ABC Company merged with StarY Company. UI standards followed by the application development team have changed. Code data is changed often in the CRM[9] application of ABC Company. To improve the time to implement these changes and avoid user's errors, it is requested to add an admin's screen. This screen will be used directly by business experts without the need for IT personnel.

Use the SNAP sub-category to size the development of the new screen. Use sub-category to size the creation and the maintenance of the code data.

7.2.7 Sub-category 2.2 help methods

7.2.7.1 What to check

Check the help object.

7.2.7.2 Rules

Count SP for any types of “help” that are not FP counted: static text, static web pages[10], tool tips, dynamic help on mouse over (context help) help window (accessed through a function key). See example “Help application” in ISO/ IEC 20926:2009 (IFPUG FSM) for FP-counted Help.

An EP may have functional and non-functional help types—count functional help per ISO/IEC 20926:2009 rules at the EP level, and the non-functional help at the application level.

EXAMPLE 1 A photo editing software application “ZoomX” provides photo editing options, which are either available free of cost to users or under a paid license. The requirement is to add tool tips to the existing photo editing tool icons as well as display a message about the corresponding tool usage on mouse hovering. SPs are counted to size this requirement for adding the tooltips.

EXAMPLE 2 Photo editing application has the pages “About ZoomX” and “Contact us,” which are text only static pages containing ZoomX’s history and company contact information respectively. SPs are counted to size these static pages.

7.2.8 Sub-category 2.3 multiple input methods

7.2.8.1 What to check

Check the EP.

7.2.8.2 Rules

a) For organizations that are using the single instance approach:

   — Count FPs for the first input method only.

   — Count SPs for each additional input method.

   — For new development requiring n input methods, count FPs for one input method and SPs for (n-1) input methods.

b) For organizations using the multiple instance approach:

   — Count FPs per for each input method.

9) Customer Relationship Management (CRM) is a strategy that companies use to manage interactions with customers and potential customers. CRM helps organizations streamline processes, build customer relationships, increase sales, improve customer service, and increase profitability.

10) A static web page is delivered to the user exactly as stored, in contrast to dynamic web pages, which are generated by a web application. Consequently, a static web page displays the same information for all users, from all contexts.

— Do not count SPs.

EXAMPLE A health care claim insurance software application receives claim inputs via web screens (maintained by another application outside the boundaries) or via batch inputs. Organizations using single instance approach count claim input via the screen (first instance of same input type) using FPs. All other similar inputs, like batch input, would be counted using SPs.

Organizations using multiple instance approach already accounted for this complete functionality using FPs. Thus, SPs cannot be applied in those cases.

7.2.9 Sub-category 2.4 multiple output methods

7.2.9.1 What to check

Check the EP.

7.2.9.2 Rules

If the organization is using the single instance approach:

1) Count FPs for the first output method only.

2) Count SPs for each additional output method.

3) For new development requiring n output methods, count FPs for one output method and SPs for (n – 1) output methods. This applies only in case that they are identical EP that don't meet the definition of uniqueness in ISO/IEC 20926:2009.

If the organization is using the multiple instance approach:

1) Count FPs per for each output method.

2) Do not count SPs.

EXAMPLE The health care claim insurance application sends the processed claim output report via web screens, as a paper output or as a downloadable PDF. Organizations that use the single instance approach count the claim report output via the screen (first instance of same output type) using FPs, but all other similar outputs would be counted using SPs. Organizations that use the multiple instance approach have already accounted for this complete functionality using FPs. SPs are not applied in those cases.

7.2.10 Sub-category 3.1 multiple platforms

7.2.10.1 What to check

Check the EP.

7.2.10.2 Rules

a) For n platforms of any category (as defined in 5.3.1) while n > 1, count SPs per 5.3.1.

b) If n = 1, do not count SPs.

EXAMPLE 1[11]: For a banking application written in JAVA, the "my account view" transaction is required to be available on three different browsers—Edge, Chrome and Firefox in the same format. The first platform deployment is FPs counted; SPs are counted for required transaction compatibility on the two other platforms, using this category.

11) The following information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE, ISO or IEC of these products. Equivalent products may be used if they can be shown to lead to the same results.