BS ISO/IEC/IEEE 32430:2025

BSI Standards Publication

Software engineering — Software non-functional size measurement

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 32430:2025

National foreword

This British Standard is the UK implementation of ISO/IEC/IEEE 32430:2025. It supersedes BS ISO/IEC/IEEE 32430:2021, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/15, Software and systems engineering.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2025
Published by BSI Standards Limited 2025

ISBN 978 0 539 26835 5

ICS 35.080

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 28 February 2025.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 32430:2025

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)

© IEEE 2025

Documents
Document Sharing Hub

Full document details: BS ISO/IEC/IEEE 32430:2025

COPYRIGHT PROTECTED DOCUMENT

© IEEE 2025
All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO or IEEE at the respective address below or ISO's member body in the country of the requester.

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

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

Contents

Page

Foreword ..... **v**

Introduction ..... **vi**

1 Scope **1**
1.1 Overview 1
1.2 Purpose 1
1.3 Word usage 1

2 Normative references **2**

3 Terms, definitions and abbreviated terms **2**
3.1 Terms and definitions 2
3.2 Abbreviated terms 8

4 Introductory information **8**
4.1 User requirements for a system 8
4.2 Non-Functional Size Measurement (NFSM) introduction 9
4.3 Software-intensive system and software product 10
4.4 Software domains 10
4.5 The relations between non-functional requirements (NFR) and functional user requirements (FUR) 10
4.5.1 Non-functional requirements 10
4.5.2 The relations between NFR and SNAP sub-categories 10
4.6 Current classification and Future evolution of NFR 13
4.6.1 The challenge 13
4.6.2 Current classification of NFR 13
4.6.3 Sizing quality-in-use requirements 13
4.7 Objectives and benefits 14
4.7.1 Objectives 14
4.7.2 Benefits 14

5 Non-functional size: Categories and sub-categories **15**
5.1 Category 1: Data operations 15
5.1.1 Sub-category 1.1: Data entry validation 15
5.1.2 Sub-category 1.2: Logical and mathematical operations 16
5.1.3 Sub-category 1.3: Data formatting 18
5.1.4 Sub-category 1.4: Internal data movements 19
5.1.5 Sub-category 1.5: Delivering added value to users by data configuration 21
5.2 Category 2: Interface design 23
5.2.1 Sub-category 2.1—User interfaces 23
5.2.2 Sub-category 2.2—Help methods 25
5.2.3 Sub-category 2.3—Multiple input methods 28
5.2.4 Sub-category 2.4—Multiple output methods 29
5.3 Category 3: Technical environment 31
5.3.1 Sub-category 3.1: Multiple platforms 31
5.3.2 Sub-category 3.2: Database technology 33
5.3.3 Sub-category 3.3: Batch processes 35
5.4 Category 4: Architecture 36
5.4.1 Sub-category 4.1: Component-based software 36
5.4.2 Sub-category 4.2—Multiple input/output interfaces 37
5.5 Sizing code data 41
5.5.1 Code data characteristics 41
5.5.2 Handling code data from non-functional sizing perspective 42
5.5.3 How code data is sized using SNAP 42

6 The sizing process **43**
6.1 Introduction 43
6.2 The timing of the non-functional sizing 44
6.3 Non-functional sizing and FSM 44

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

6.4 Steps to determine the non-functional size 45
6.4.1 Step 1: Gather available documentation 45
6.4.2 Step 2: Determine the sizing purpose, type, scope, boundary, and partition 45
6.4.3 Step 3: Identify the NFR 48
6.4.4 Step 4: Associate NFR with sub-categories and identify the SCU 49
6.4.5 Step 5: Determine the SNAP size for each sub-category 49
6.4.6 Step 6: Calculate the non-functional size 49
6.4.7 Step 7: Document and report 49
6.5 Calculating the non-functional size 50
6.5.1 Formula approach 50
6.5.2 Determine the non-functional size of each sub-category 50
6.5.3 Determine the non-functional size of a development project 50
6.5.4 Determine the non-functional size of an enhancement project 50

7 Complementarity of the functional and the non-functional sizes 53
7.1 General 53
7.2 Requirements involving functional and non-functional requirements 53
7.2.1 Sub-category 1.1 data entry validation 53
7.2.2 Sub-category 1.2 logical and mathematical operations 54
7.2.3 Sub-category 1.3 data formatting 55
7.2.4 Sub-category 1.4 internal data movements 56
7.2.5 Sub-category 1.5 delivering added value to users by data configuration 56
7.2.6 Sub-category 2.1 user interfaces 57
7.2.7 Sub-category 2.2 help methods 58
7.2.8 Sub-category 2.3 multiple input methods 58
7.2.9 Sub-category 2.4 multiple output methods 59
7.2.10 Sub-category 3.1 multiple platforms 59
7.2.11 Sub-category 3.2 database technology 60
7.2.12 Sub-category 3.3 batch processes 60
7.2.13 Sub-category 4.1 component-based software 60
7.2.14 Sub-category 4.2 multiple input/output interfaces 61

Annex A (informative) **NFSM strengths 62**

Annex B (informative) **Use of non-functional size 63**

Bibliography 70

IEEE Notices and Abstract 72

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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.

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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.

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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.

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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:

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

— 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.]

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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.]

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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).

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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]

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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

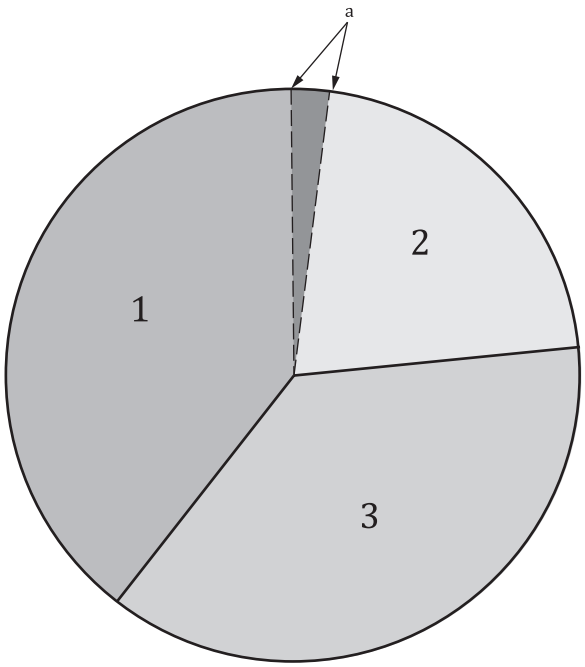
4 Introductory information

4.1 User requirements for a system

Figure 1 illustrates how requirements can be classified.

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025



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.

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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.

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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.

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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).

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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;

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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).

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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.

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

"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

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025

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).

© IEEE 2025 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC/IEEE 32430:2025