BRITISH STANDARD

BS ISO/IEC 24570:2005

Software engineering — NESMA functional size measurement method version 2.1 — Definitions and counting guidelines for the application of Function Point Analysis

ICS 35.080

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

BSI Documents
Document Sharing Hub
British Standards

Full document details: BS ISO/IEC 24570:2005

National foreword

This British Standard reproduces verbatim ISO/IEC 24570:2002 and implements it as the UK national standard.

The UK participation in its preparation was entrusted to Technical Committee IST/15, Software engineering, which has the responsibility to:

— aid enquirers to understand the text;

— present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;

— monitor related international and European developments and promulgate them in the UK.

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

Cross-references

The British Standards which implement international publications referred to in this document may be found in the *BSI Catalogue* under the section entitled "International Standards Correspondence Index", or by using the "Search" facility of the *BSI Electronic Catalogue* or of British Standards Online.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the ISO/IEC title page, pages ii to viii, pages 1 to 124, an inside back cover and a back cover.

The BSI copyright notice displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 16 March 2004

© BSI 16 March 2004

ISBN 0 580 45602 1

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

INTERNATIONAL STANDARD

ISO/IEC 24570

First edition
2005-02-15

Software engineering — NESMA functional size measurement method version 2.1 — Definitions and counting guidelines for the application of Function Point Analysis

Ingénierie du logiciel — Méthode de mesure de la taille fonctionnelle NESMA, version 2.1 — Définitions et manuel des pratiques de comptage pour l'application de l'analyse des points fonctionnels

Reference number
ISO/IEC 24570:2005(E)

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24570:2005

Contents

Page

Foreword v
Introduction vi
1 Scope 1
2 Overview 1
2.1 Objective of this International Standard 1
2.2 Focus of this International Standard 1
2.3 Organization of this International Standard 2
3 Introduction to FPA 3
3.1 Brief description of FPA 3
3.2 Use of FPA: application function point count versus project function point count 4
3.3 The types of function point counts 5
3.4 Function point counts during a project 5
3.5 Scope of the count and boundary of the application to be counted 5
3.6 Users 5
3.7 Functions and function types 6
3.8 The complexity of a function 6
3.9 The valuing of function types 7
3.10 The function point count 7
4 Guidelines to carry out an FPA 7
4.1 Step-by-step plan for carrying out an FPA 8
4.2 Types of function point counts and their accuracy 8
4.3 The role of the quality of the specifications 10
4.4 FPA during a project 11
4.5 Determining the application function point count 11
4.6 Determining the project function point count 13
4.7 FPA in specific situations 16
4.8 Illustration: FPA and the system life cycle 20
5 General counting guidelines 25
5.1 Counting from a logical perspective 25
5.2 Applying the rules 25
5.3 Built functionality, not requested functionality 25
5.4 Double counting 25
5.5 Production of re-usable code 26
5.6 Re-use of existing code 26
5.7 Screens and reports 26
5.8 Input and output records 26
5.9 Security and authorization 26
5.10 Operating systems and utilities 26
5.11 Report generators and query facilities 27
5.12 Graphs 27
5.13 Help facilities 27
5.14 Error messages and other messages 27
5.15 Menu structures 28
5.16 List functions 28
5.17 Browse and scroll functions 28
5.18 Cleaning functions 28
5.19 Completeness check on the function point count 29
5.20 FPA tables 29

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

5.21 Deriving logical files (data functions) from a normalized data model 30
5.22 Shared use of data 34

6 Internal Logical files 35
6.1 Definition of an internal logical file 35
6.2 Counting internal logical files 36
6.3 Determining the complexity of internal logical files 37

7 External Interface Files 38
7.1 Definition of an external interface file 38
7.2 Counting external interface files 38
7.3 Determining the complexity of external interface files 40

8 External inputs 40
8.1 Definition of an external input 41
8.2 Counting external inputs 42
8.3 Determining the complexity of external inputs 44

9 External Outputs 45
9.1 Definition of an external output 45
9.2 Counting external outputs 47
9.3 Determining the complexity of external outputs 50

10 External inquiries 51
10.1 Definition of an external inquiry 51
10.2 Counting external inquiries 52
10.3 Determining the complexity of external inquiries 53

11 Practical Situations and their solutions 54
11.1 Standard authorization functions 55
11.2 Specific authorization functions 55
11.3 Report generator and query facility 56
11.4 Help functions 56
11.5 Error messages 57
11.6 Menu structures 57
11.7 FPA tables 58
11.8 Denormalization 59
11.9 Counting logical files (data functions) 61
11.10 Combined external inputs 65
11.11 Counting a transaction file 66
11.12 Reports on different media 67
11.13 Daily and weekly processing 69
11.14 Conversion 69
11.15 External outputs with summary information 70
11.16 The number of data elements on a report 72
11.17 Combined external outputs 72
11.18 Combination effects with functions 77
11.19 Querying with several search keys 79
11.20 Screens with list function 81
11.21 Browse and scroll functions 82
11.22 Selection screens and changing data with a search key 85
11.23 Direct and delayed processing 89
11.24 Case study of a customer application 91

Annex A (normative) **The most important features and tables for valuing function types** 95

Annex B (normative) **Function point analysis glossary** 101

Annex C (informative) **Application of function point analysis including general system characteristics** 106

Annex D (informative) **General system characteristics** 113

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 24570 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and system engineering.*

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Introduction

Version 1.0 (November 1990)

The NESMA Board set up a counting guidelines committee devoted to the standardization of counting guidelines/definitions in September of 1989. The committee's task was and (still) is to draw up and maintain a NESMA FPA manual.

Version 1.1 (May 1991)

Version 1.1 is a reprint of version 1.0. Except for the improvement of some minor errors, the two versions are the same.

Addendum (May 1994)

The manual *Definitions And Counting Guidelines For The Application Of Function Point Analysis* satisfies a great need and has become a standard in the Netherlands within a short time.

In May of 1991 the Board of the NESMA set up the work group "FPA Case Study" and gave it the task of developing a case study that would present the application of FPA and counting guidelines within a context.

While developing the case study, the work group felt that a number of definitions of counting guidelines needed to be more precise:

- The derivation of logical files from a data model in third normal-form (the so-called denormalization rules)
- A more concrete definition of the concept of FPA table
- Uniform treatment of selection screens
- Dealing with combination effects of functions

The Counting Guidelines Committee established additional counting guidelines for these topics after extensive discussion took place both within the committee itself and within the work group FPA Case Study.

You will find the additional counting guidelines necessary and/or helpful when working out the case. In view of the issue date of the case (mid 1994), the NESMA Board decided to issue these additional counting guidelines as an Addendum to version 1.1 of the Counting Guidelines Manual.

Version 2.0 (April 1996)

This new version of the manual *Definitions And Counting Guidelines For The Application Of Function Point Analysis* incorporates the following improvements:

- The guidelines recorded in the addendum have now been integrated into the manual
- A large number of points in the guidelines have been further clarified
- The results of extensive consultation with the IFPUG have been processed
- The manual's accessibility has been increased further as a result of editorial improvements
- Many examples and illustrations have been added

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24570:2005

The committee is of the opinion that the changes made are chiefly an elaboration and further illustration of the guidelines drawn up earlier. In modifying the manual, the committee has worked in such a way that the changes made have as little effect as possible on the results of a function point analysis. Appendix D goes further into this.

The guidelines published in this manual have been applied to a rather large case study with the title, *FPA Case Study "Hotel" For The Application of Function Point Analysis*. Applying the guidelines in practice is explained in this document in detail.

The publication of this version takes precedence over versions 1.0 and 1.1, as well as the Addendum.

English translation of version 2.0 (November 1997)

This English version of the manual is an accurate translation of the Dutch version

Version 2.1 Unadjusted (February 2002)

This version has been developed for the manual to be an ISO recognized standard. The main adaptation is the exclusion of the General System Characteristics. This exclusion conforms to the ISO standard 14143-1 Functional Size Measurement.

Reason for this International Standard

The NESMA was set up in the spring of 1989. (At that time it was called the NEFPUG.) During its first meeting in June, it carried out a study among its participants in order to survey which subjects they were interested in. The standardization of counting guidelines/definitions was high on the list. In reaction to this, the NESMA Board decided to set up a committee devoted to this topic. This committee set itself the task of putting together a International Standard for the theoretical application and the practical use of function point analysis (FPA)[1].

Over the years a number of "dialects" have arisen for FPA. These dialects complicate the goal of determining the number of function points and make it almost impossible for organizations to compare results. One insufficiently acknowledged reason for this is that different interpretations of the "Albrecht" method have arisen.

This International Standard hopes to provide clarity by formulating standards for the definitions and counting guidelines that pertain to FPA.

Intended audience

This International Standard is meant for everyone who performs function point counts; i.e., both for people who count according to the NESMA rules and for those who use the IFPUG rules. For those using the IFPUG rules, the NESMA International Standard can be a valuable supplement to the IFPUG International Standard if the differences stated on the website “WWW.NESMA.ORG” are taken into account. The NESMA International Standard, after all, contains many hints, guidelines, and examples that can be of value to every FPA counter. It is assumed that the reader has some knowledge of FPA. Nevertheless, we have also attempted to produce as complete a International Standard as possible that includes sufficient introductory material and explanation for the new FPA user. For both the maintenance of the IFPUG International Standard and the NESMA International Standard there is a co-operation between the IFPUG CPC and the NESMA CPC.

Departure points of this International Standard

The NESMA FPA method is in principle applicable to all Functional domains.

The following documentation has served as the foundation for this International Standard:

- IBM CIS & A Guideline 313, AD/M Productivity Measurement and Estimate Validation, November 1, 1984.

---

[1] The abbreviation FPA is used for the term Function Point Analysis.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

This is an internal IBM publication. The method described in it is usually referred to as *Albrecht '84.*

Future versions

When changes and supplements to this International Standard prove necessary in the future, an entire new version will be produced

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 24570:2005

Software engineering — NESMA functional size measurement methode version 2.1 — Definitions and counting guidelines for the application of Function Point Analysis

1 Scope

This International Standard:

a) specifies a method to measure the functional size of software,

b) gives guidelines on how to determine the components of functional size of software,

c) specifies how to calculate the functional size as a result of the method, and

d) gives guidelines for the application of the method.

2 Overview

This clause provides an overview to the International Standard "Definitions and counting guidelines for the application of function point analysis". The following questions are answered: What is its aim (subclause 2.1)? What is its focus (subclause 2.2)? How is it laid out (subclause 2.3)?

2.1 Objective of this International Standard

The International Standard attempts to provide a theoretical framework by presenting definitions and standard guidelines. It also tries to illustrate the counting guidelines as concretely as possible by using several practical situations.

2.2 Focus of this International Standard

The International Standard focuses on how the functional size of an application is determined. The International Standard does not go into any of the aspects that play a role when project budgeting is drawn up on the basis of this functional size; e.g., productivity standards and productivity attributes. This particular topic has been described in the manual "*Budgeting on the basis of logical design using function point analysis",* also by the NESMA.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Figure 1 indicates what this International Standard *will* and *will not* cover.

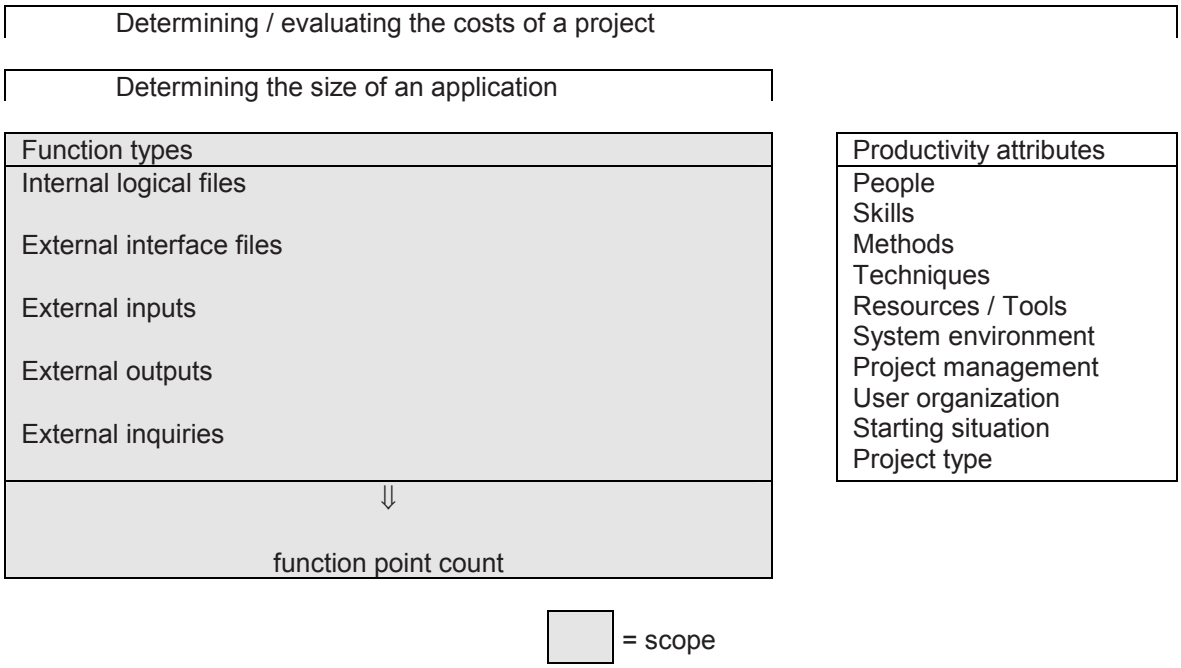

Figure 1 — Scope of the International Standard

2.3 Organization of this International Standard

The terms and concepts used in the International Standard are explained in Clause 2 and defined in Annex B.

Clause 3 provides an introduction to FPA and in which the functional aspect of FPA is emphasized. It will also spell out briefly what FPA is and explains the terms that form the basis for the concept of FPA. Matters such as distinguishing between an application function point count and a project function point count are examined, just as are other various types of function point counts, the role of FPA during a project, users, and function point count.

Clause 4 provides you with an overview of the position of FPA in a project and with the types of function point counts that can be carried out during the life cycle of an application. In other words, the chapter will explain when points can be counted and what information is needed minimally in order to count. The chapter will also give a step-by-step plan for carrying out a function point analysis and indicates how projects, applications, and packages should be counted. Each of these requires their approach.

Clause 5 states general counting guidelines for a function point count.

Clauses 6, 7, 8, 9 and 10 give successively the definitions and guidelines used to identify function types and to determine the complexity of function types for internal logical files, external interface files, external inputs, external outputs, and external inquiries. The guidelines are broken down per function type for identifying the function type concerned, for determining the number of data element types, and for determining the number of record types or referenced logical files.

Clause 11 provides several practical situations and their solutions. The counting guidelines are not repeated explicitly here, but reference is made to the relevant guideline(s) or section(s) on which a solution is based.

Annex A is meant to be a short summary of the guidelines and contains the most important features of each function type, as well as the tables for valuing the function types.

Annex B contains the definitions of the terms that this International Standard uses.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

For backward compatibility with previous adjusted function point counts, Annex C describes the application of FPA with the general system characteristics that lead to a value adjustment factor with which the adjusted function point count can be determined from the function point count.

Annex D describes the general system characteristics.

The Alphabetic index of keywords is given after the table of contents.

This International Standard has been set up in such a way that the reader does not necessarily have to start at Clause 1 before continuing on to Clause 2, then 3 and 4 etc. Instead, he can look up what he thinks is important to him. For one person, specific counting guidelines for a particular function type may be important, while someone else may want a more general frame of reference for an initial introduction to FPA.

3 Introduction to FPA

This chapter gives a short description of FPA and explains a number of important concepts related to it. More specifically, section 3.1 provides a brief synopsis of FPA. Sections 3.2 through 3.4 distinguish between the different kinds of function point counts. Sections 3.5 through 3.9 discuss each of the following successively within the context of FPA:

- The boundaries in which counting takes place
- Users
- Function types
- The complexity of a function type
- The valuing of function types

Section 3.10 defines the term " function point count" and describes how it is determined.

3.1 Brief description of FPA

3.1.1 Background, purpose and application of FPA

FPA was developed by A.J. Albrecht at IBM between 1974-1979 as a result of productivity research into a large number of projects. The first version of FPA was introduced in 1979, followed by adaptations based on practical experiences in 1983 and 1984.

FPA is currently applied in countless organizations throughout the entire world. Experiences with the technique are exchanged in user groups: the International Function Point Users Group (IFPUG), the Australian Software Metrics Association (ASMA), the United Kingdom Software Metrics Association (UKSMA), the NESMA, and various other national user groups.

FPA introduces a unit, the function point, to help measure the size of an application that is to be developed or maintained. The word "application" within the framework of FPA means "an automated information system". The function point expresses the quantity of information processing that an application provides a *user*. This unit of measurement is separate from the way in which the information processing is realized in a technological sense. A function point is an abstract term and can be compared somewhat to so-called "rental points". Rental points are based on the number of rooms in a house, the surface area of these rooms, the number of facilities the house has, and the location of the house. This then serves as a measurement for a residence offered to a potential tenant.

FPA was first used to *measure the productivity* of system development and system maintenance after an application was built. It soon became clear that the technique could also be used to support *project budgeting* because the data needed for an FPA can be made available early on in a project.

This FPA method may be applied to all functional domains.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

3.1.2 Rationale behind FPA

The three separate words that make up the term "Function Point Analysis" can be used to explain the way of thinking behind FPA.

Function

As mentioned earlier, FPA bases itself on the functionality that an application provides a *user* (see section 3.6). Because users see only the "outside" or the *boundary* (see section 3.5) of an application, FPA examines the specifications that describe the application's exchange of information with its environment. Functionality is derived from incoming and outgoing information flows (these can be both data or control information), as well as from the logical files that an application contains or uses. The functionality of an application is measured by determining the *function types*. (See section 3.7.)

Point

The *complexity* of each function type is determined according to certain standard guidelines. (See section 3.8.) Each function is worth a number of points, depending on its complexity (section 3.9). The sum of these points yields the number of function points (section 3.10. This is the basis for project budgeting.

Analysis

FPA is the analysis of an application or the analysis of the description/specification of an application in order to establish its number of function points. The act of establishing the quantity of an application's function points is also called *function point counting*.

In order to be able to carry out a function point count the following must first be determined:

- Purpose of the function point count (section 2.2)
- Scope and boundaries of the application or project to be counted (section 2.5)

This concludes a summary of the methodology and a brief description of FPA. The sections that follow explain the various terms used in FPA.

3.2 Use of FPA: application function point count versus project function point count

Function point counts can be linked to applications or to projects. This means that a distinction is made between the following two objectives:

- Determining the application function point count
- Determining the project function point count

Application function point count:

> The number of function points that measures the amount of functionality that an application is to supply or has already supplied to a user. It also measures the size of an application that must be maintained.

Project function point count:

> The number of function points that measures the amount of functionality of a new application or of changes to an existing application. Changes to an existing application pertain to adding, changing, and deleting functions. The project function point count is an essential parameter when determining the effort and lead-time required for a project.

Determining the application function point count is elaborated on in section 4.5. Section 4.6 discusses the project function point count further.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

3.3 The types of function point counts

One of three kinds of function point counts can be chosen, depending on the degree of detail of the specifications available. The following represent the different types of function point counts. Note that they are listed by degree of detail, number one having the least detail and number three the most:

1. Indicative function point count
2. Estimated function point count
3. Detailed function point count

These function point counts are explained further in section 4.2.

3.4 Function point counts during a project

Function point counts can be carried out at different times during a project. They can therefore be related to the phases of a project; e.g., the planning phase, the execution phase, and the evaluation phase. As a result, the following breakdown of function point counting arises: the *initial function point count*, the *interim function point count*, and the *final function point count*. These counts are discussed further in section 4.4.

3.5 Scope of the count and boundary of the application to be counted

The scope of the count is the set of functional requirements/specifications to be included in the function point count. When you have determined the scope you can define the boundary, the conceptual interface between the application and its users.

As indicated earlier in section 3.1, the scope of the count and the boundary of an application to be counted plays an important role in FPA. Consequently, the boundaries of the application to be counted must first be determined in order to be able to carry out a function point count.

The boundary is necessary in order to be able to determine:

- The application that certain data belongs to
- Which data crosses the boundary

As mentioned in section 3.2, a distinction is made between a function point count for an application and a function point count for a project. Section 4.5.1 provides guidelines for determining the application function point count and section 4.6.1 gives guidelines for determining the project function point count.

3.6 Users

FPA acknowledges three kinds of users:

- The people and/or organizations that use or are going to use the application to be measured. This category includes the following: end-users, functional managers, and operators.
- The owner and/or employee(s) who determine(s) the requirements and wishes included in the specifications. These requirements and wishes are recorded on the basis of the demands of the end-user(s) for example, but also on the basis of requirements that a government or its legislation can impose on the application.
- Other applications that use the data or the functions of the application to be counted.

Because the function point count takes place from the perspective of the user, it is always necessary to have it done in cooperation with the user or, at the very least, to have the result of the count verified by the user. The user, after all, is the only one who can determine whether a certain function is being requested.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

3.7 Functions and function types

The function point count measures the size of the functions of an application or a part of an application. The count revolves around the *what* and not the *how* of the application furnished. Only those components that the user requests, that he can recognize, and that he considers significant are assessed. These components are called functions or base functional components. A function belongs to a function type.

FPA defines *functions* as follows:

> The five types of components of which an application exists, as seen from the perspective of FPA. These components determine the amount of functionality an application provides to a user.

Function types are categorized into two main groups:

- Data function types
- Transactional function types

A *data function type* is:

> A logical group of data seen from the perspective of the user.
>
> FPA distinguishes between:
>
> - Internal logical files
> - External interface files

A *transactional function type* is:

> A succession of actions that the user sees as a single work unit. FPA distinguishes between:
>
> - External inputs
> - External outputs
> - External inquiries

Each function type is discussed further in chapters 5 through 9.

3.8 The complexity of a function

The *complexity of a function* is defined as follows:

> The weight of a function on the basis of which a number of function points are allocated to the function.

The complexity of a function is determined by using the appropriate complexity matrix. A separate table has been defined for each function type. Complexity depends on the number of data element types and the number of referenced logical files connected to a given function. Three levels of complexity are distinguished:

Low: Few data element types and/or referenced logical files are involved with the function

Average: The function is neither low nor high as regards complexity

High: Many data element types and/or referenced logical files are involved with the function

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

3.9 The valuing of function types

After the complexity of a function has been determined as described in chapters 5 through 9, the number of function points can be allocated to the function. This should be done according to the rating illustrated in the table below.

Table 1 — Function point table for determining the value of function types

	ILF	EIF	EI	EO	EQ
Low	7	5	3	4	3
Average	10	7	4	5	4
High	15	10	6	7	6

ILF = Internal logical file

EIF = External interface file

EI = External input

EO = External output

EQ = External inquiry

Specifications are enough to identify functions and their type when carrying out an *estimated* function point count (see section 4.2.2), but it will be difficult to determine the complexity of these functions. In such a case, a data function is rated as *low*, while the rating *average* is used for a transactional function.

3.10 The function point count

The function point count is the sum of the number of function points assigned to each of the functions (in the way described above) that lie within the boundaries of the object to be counted; i.e., the application or the project.

A NESMA-FSM-method measurement result on the FUR/specifications for a piece of software shall be labeled according to the following convention:

F(unction) P(oint) (ISO/IEC 24570:2003, NESMA FSM method V2.1).

The values for the function types are defined in Annex A.

4 Guidelines to carry out an FPA

This chapter indicates how function point analysis should be carried out. To this end, section 4.1 will first present a generally applicable step-by-step plan. Section 4.2 will then indicate how you should act when dealing with an indicative, estimated, and detailed function point count. Section 4.3 goes into the role of the quality of specifications, while section 4.4 explains the use of FPA during a project. Sections 4.5 and 4.6 show how an application and a project function point count are determined in the event of development and in the event of enhancement, respectively. Section 4.7 states what you must take into consideration when dealing with the different ways of recording specifications. Section 4.8 concludes the chapter with an illustration of how the different types of function point counts can be applied during the life cycle of an application. For illustration purposes, this section will assume a general system life cycle as a phasing method for the life cycle of an application.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.1 Step-by-step plan for carrying out an FPA

Below follows a step-by-step plan to perform a function point count.

Step 1: Collect the available documentation. The documentation that should be present for an indicative function point count, an estimated function point count, and a detailed function point count is described in sections 4.2.1, 4.2.2, and 4.2.3, respectively.

Step 2: Determine who the users of the application are. (See section 3.6.) Determine from which other application(s) the application to be counted receives and/or uses data.

Step 3: Establish whether an application function point count or a project function point count must be carried out. If an application function point count must be performed, follow the instructions stated in section 4.5. If a project function point count must be performed, follow the instructions found in section 4.6.

Step 4: Identify the functions and determine their type and complexity according to the guidelines described in chapters 5 through 9. When doing so, adhere to the sequence in which the chapters appear. Assign the number of function points using the function point table illustrated in section 3.9. This will result in the function point count.

Register the structure of the count and the number of function points. Particularly record any departure points and the assumptions that have been made.

Step 5: Together with the user(s), verify the result in relation to those aspects where specific interpretation of the available *specifications* was needed. If necessary, make any corrections as a result of that verification.

Step 6: Verify the result with an FPA expert in relation to those aspects where specific interpretation of the *counting guidelines* was needed. This may or may not be necessary. Make any corrections that are required as a result of that verification.

4.2 Types of function point counts and their accuracy

Depending on the degree of detail of the specifications available, one of three types of function point counts can be chosen: *an indicative, an estimated, or a detailed function point count*. These three types of function point counts can be carried out when both an application function point count and a project function point count is being determined. The minimum specifications required are different for each of the three types of function point counts. In the sections below, the specifications required to carry out each of the three counts are stated. Each section, finally, will indicate when a particular type of count can be executed in the life cycle of an application.

4.2.1 The indicative function point count

Definition

An indicative function point count indicates the size of an application or a project based on either a conceptual data model or a normalized data model. Caution is advised when using this indication as deviations of up to 50% higher or lower are possible!

If a conceptual data model is assumed, the number of function points is equal to:

The number of entity types of the internal logical file type
*in the conceptual data model * 35*
+
the number of entity types of the external interface file type
*in the conceptual data model * 15*

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Note that the entity types that are from the FPA table type (see section 5.20) *and* that are maintained by the application to be counted are counted together as one entity type. The same applies to the entity types that are from the FPA table type but which are maintained by a different application. For FPA tables, therefore, two entity types are counted maximally.

The multiplication factors are based on the assumption that a minimum of three external inputs (add, change, and delete), one external output, and one external inquiry will usually be present for each internal logical file, and that a minimum of one external output and one external inquiry will be present for each external interface file.

When a data model in third normal-form is available, the following formula indicates the number of function points:

The number of entity types of the internal logical file type
*in the normalized data model * 25*
+
the number of entity types of the external interface file type
*in the normalized data model * 10*

Here, too, the entity types that are from the FPA table type (see section 5.20) *and* that are maintained by the application to be counted are counted together as one entity type. The same applies to the entity types that are from the FPA table type but which are maintained by a different application. For FPA tables, therefore, two entity types are counted maximally.

Applicability

Given a general system life cycle as a phasing method for the life cycle of an application, an indicative function point count can almost always be carried out at the end of that system life cycle's requirements phase. In a large number of cases, the indication can even be obtained after the information planning phase has been completed.

The specifications required

The above shows that you should have the following at your disposal for an indicative function point count:

- A conceptual or normalized data model of the application to be counted
- An indication as to how the logical files distinguished are maintained: by the application to be counted or by a different application

4.2.2 The estimated function point count

Definition

An estimated function point count determines the number of functions for each function type (transactional function types and data function types) and uses a standard value for complexity: *Average* for transactional function types and *Low* for data function types.

Applicability

The moment at which an estimated function point count can be carried out depends to a large extent on when certain specifications are provided. This, again, depends on the method used for system development. Given a system life cycle of an application that contains the analysis phase, this moment would be at the end of the analysis phase.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

The specifications required

The following specifications must be available for an estimated count:

- A (structure) model that shows the logical files involved and their relations
- An indication as to how the logical files distinguished are maintained: by the application to be counted or by a different application
- A model that shows the computer system functions with their incoming and outgoing information flows
- The information flows between the functions of the application to be counted and its environment

4.2.3 The detailed function point count

Definition

The detailed function point count is the most accurate count in which all the specifications needed for FPA are known in detail. This means that transactional function types have been specified up to the level of referenced logical files and data element types, and that logical files have been specified up to the level of record types and data element types. As a result, the complexity for each function identified can be established.

Applicability

The moment at which a detailed function point count can be carried out depends on the phasing method used for the life cycle of an application. Given a system life cycle of an application that contains the functional design phase, this moment would be during functional design or at the end of functional design, when specifications are available with a sufficient degree of detail.

The specifications required

The following specifications must be available for a detailed count:

- A model with all logical files and their relations; e.g., a Bachman diagram or an Entity Relationship Diagram
- The record types and the data element types of the logical files
- An indication as to how the logical files distinguished are maintained: by the application to be counted or by a different application.
- A model that shows the computer system functions, their incoming and outgoing information flows, the logical files involved with the functions, and the supporting functions (help functions and so on).
- A detailed specification of the incoming and outgoing information flows of the application up to the level of data element types.

4.3 The role of the quality of the specifications

Regardless of the type of function point count (as described in section 4.2), sound functional specifications of the application are needed in order to carry out a function point count. The easiest way to draw up these specifications is to work in a standard and methodical fashion. More and more, software developers have the tendency to adopt the use of generally accepted methods. Depending on the method used, the form of the specifications produced can differ. The function point count, however, should continuously render the same result for the same application.

An FPAs reliability depends directly on the quality of the specifications provided. Good-quality specifications will ensure that little effort is needed to translate specifications into function points and will result in a reliable count. Flawed or incomplete specifications can make it impossible to carry out an FPA, or the result of the count may differ considerably from person to person as a result of various interpretations of the specifications.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.4 FPA during a project

FPA plays a role during the entire course of a system development project. For instance, during the planning phase of a project, an *initial function point count* is carried out. As soon as change requests pertaining to specifications already recorded have been submitted during the course of a project, *interim function point counts* are carried out. When a project has been fully completed, the *final function point count* determines how big the supplied product is and what the project function point count has been. It does not matter which phases of an application's life cycle are being carried out in the project. Given a system life cycle as a phasing method, for example, the project can pertain to analysis, functional design, and construction. Regardless of the projects content, there will always be an initial and a final function point count, and sometimes one or more interim function point counts.

An *initial function point count* is:

> a count at the beginning of a project for developing an application or for changing an application (adding, changing, and deleting functionality) in which an application function point count (see section 4.5) and a project function point count (see section 4.6) is recorded.

An initial function point count should be carried out at the beginning of a project. Depending on the specifications available, this count is either an indicative, estimated, or detailed function point count. The objective of the initial function point count is to draw up a budget for the project in terms of effort and lead-time.

An *interim function point count* is:

> a count during a new development project or an enhancement project in which the size of an interim enhancement (addition, change, or deletion of functionality) is determined. This means that the influence that a change will have on both an application function point count (see section 4.5) and a project function point count (see section 4.6) is recorded.

An interim function point count should be done when functional specifications are changed. Depending on the specifications available, this count is an indicative, estimated, or detailed function point count. The objective of the interim function point count is to determine the influence of change requests on the price and delivery date agreed upon with the customer.

A *final function point count* is:

> a count at the end of a project for developing an application or for changing an application (adding, changing, or deleting functionality) in which the final application function point count (see section 4.5) and the final project function point count (see section 4.6) are recorded.

A final function point count takes place at the end of a project. The project can pertain to the development of an application or to the realization of enhancements during an application's operation and maintenance phase. One objective of the final function point count is to determine the size of the *application* in function points. Another objective is to determine the size of the *project* so that productivity can be determined and, for example, a final monetary settlement for the project can take place, if a fixed price per function point has been agreed upon.

4.5 Determining the application function point count

As stated in section 3.2, FPA can be used to determine the application function point count or the project function point count. This section explains the use of FPA when determining the *application* function point count.

The purpose of carrying out counts on applications is to determine the amount of functionality that is to be furnished to a user or that has already been furnished to him. This means that the functional specifications of an application must serve as the starting point and *not* the physical components such as programs or physical files.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Determining the application function point count can be carried out differently when developing an application than when enhancing an application. Still, in both cases, the application boundary must first be determined. This and the counting method employed for both development and enhancement are covered in greater depth in the sections below.

4.5.1 Determining the application boundary

An *application boundary* is:

> the border between the application and its environment (other applications and users). This border of course marks the scope of the application function point count.

An *application* in this International Standard refers to:

> an automated information system. This is an application that collects, saves, processes, and presents data by means of a computer.

The following guidelines can help in determining the application boundary:

- The application demarcated by the application boundary should make up an independent whole that can function separately from other applications to a large degree.
- Establish whoever the owner or main user is. If there are several owners or chief users, it often means that you are dealing with several applications.
- Look at the application through the eyes of the user; i.e., only at the part of the application the user can actually observe. In putting yourself in the user's shoes, use the specifications that describe or define the outside of the application. This is called the application context, and it can be represented in a context diagram, among other ways. Determine what is located inside and outside the application. Only those things that the user requests and that are relevant to him are significant to the function point count.
- Think of an application as a group of programs maintained as a whole. The application boundary encloses this group of programs. All functions within this boundary are counted and recorded as a whole.

4.5.2 The application function point count of new applications

This pertains to an application function point count of applications in the process of being built or that have already been built at the request of a user or user organization, and that provide a solution to the needs or wishes of the user or user organization.

Determining the application function point count during the development of an application occurs as indicated in section 4.1. If the application in a development project is realized in a single project, determining an *application* function point count does not occur differently than when determining a *project* function point count. (See section 4.6.2.) Note, however, that the size of any conversion software may not be counted when you carry out an application function point count.

If the application is being realized in the form of a number of sub-projects carried out in parallel, then the total functionality furnished by all the sub-projects will have to be examined in order to determine the application function point count. When examining these sub-projects, you should make sure that functionality appearing in more than one sub-project (such as a logical file) is not counted twice.

In the event of enhancement to an existing application (the adding, changing, or deleting of functionality), the application function point count of the (changed) application should be determined as indicated in section 4.5.3.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.5.3 The application function point count of changed applications

This sizes an application after enhancement. In principle, enhancement can take place during each phase of an application's life cycle, but usually occurs during construction or during operation and maintenance. In the event of major enhancements, a separate project can be defined in which a project function point count is determined, in addition to an application function point count. In all cases, the application function point count is determined in the same way after the enhancement.

Below are the steps to be taken in order to determine the new application function point count:

Step 1: Determine the number of function points of the application before the change (*UFPB*).

Step 2: Determine which transactions and/or logical files are deleted from the existing application and count how many function points they represent (*DEL*).

Step 3: Establish which transactions and/or logical files change. Then determine the number of function points that they represent before the change (*CHGB*) and after the change (*CHGA*).

Step 4: Identify which transactions and/or logical files are added to the application and establish how many function points they represent (*ADD*).

Step 5: Determine the application function point count of the application after the change (*AFP*) as follows:

$$AFP = [UFPB + ADD - DEL + (CHGA - CHGB)]$$

4.5.4 The application function point count of re-built applications

If one application is replaced by another with the same functionality, then the application function point count of the new application is equal to the old application it is replacing.

If enhancements are the result of such a replacement, the size of the application in function points can be determined in two ways:

- The replacement can be considered a new application. If this option is chosen, counting is done as described in section 4.5.2.
- The replacement can be considered a change to the application being replaced. In this case, counting is carried out as indicated in section 4.5.3.
- The result (the application function point count) of both counting methods is the same.

4.6 Determining the project function point count

As indicated in section 3.2, FPA can be used to determine an application function point count or a project function point count. This section explains the use of FPA to determine the *project* function point count.

Determining the size of a project (i.e., counting the number of function points of a project) differs in a number of ways from purely and simply calculating an application function point count; i.e., determining the amount of functionality provided or to be provided. This is because the effort needed does not always result in an increase of functionality. Consider, for example, an effort to remove functionality, or to create a one-time functionality in order to convert data.

Using a number of project situations, the sections below spell out how a project function point count can be determined. Figure 2 illustrates an example of how a project function point count and an application function point count are determined, as well as the differences between the two.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

		RELEASE 1	RELEASE 2	RELEASE 3
Function Points	ADD	1000	200	500
	CHANGE Before		80	220
	CHANGE After		100	200
	DELETE		40	100
Project function point count		1000	Add: +200 Change after: +100 Subtotal: +300 Delete: +40 TOTAL: 340	Add: +500 Change after: +200 Subtotal: +700 Delete: +100 TOTAL: 800
Application function point count		1000	Rel. 1: 1000 Add: +200 Change after: +100 Change before: -80 Delete: -40 TOTAL: 1180	Rel. 2: 1180 Add: +500 Change after: +200 Change before: -220 Delete: -100 TOTAL: 1560

Figure 2 — Relations between a project function point count and an application function point count

Determining the project function point count when an application is developed can be carried out differently than when an application is enhanced. Still, in both cases, the application boundary must first be fixed. This and the counting method employed for both development and enhancement are covered in greater depth in the sections below.

4.6.1 Determining the scope of a project function point count

The *scope* of a project function point count is:

> the set of functional requirements/specifications of a development project or an enhancement project to be included in a function point count. This may include one or more applications and therefore more than one application boundary may have to be determined as described in section 3.5.1

This definition shows that two kinds of projects are distinguished:

- Development projects
- Enhancement projects

A *development project* is:

> a project in which a completely new application is realized. In its execution, a development project can be split up into a number of sub-projects, each of which is responsible for a certain sub-system of the entire application. Each sub-project should then be considered an individual development project, if the sub-system itself is an application.

Re-building an existing application (re-engineering) is considered as development.

FPA defines an *enhancement project* as:

> a project in which enhancements are carried out on an existing application. This means that functionality can be added to, changed in, or deleted from an existing application.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

An *enhancement* is defined as:

> the actions performed on an application that change the specifications of that application and, as a result, usually the function point count as well. Change requests are an example of enhancement.

The following guidelines can help in determining the scope of the project:

- Look at the application to be realized through the eyes of the user. The logical structures should be taken into consideration, not the physical structures.
- An application can be developed as a number of sub-projects executed more or less in parallel, each of which realizes a sub-system. The scope of such a sub-project therefore includes a sub-system. If the sub-systems must be able to exist independently (e.g., because of a phased implementation of the application or for functional reasons), then the exchange of data between the sub-systems is included in the function point count of each sub-project. The scopey of the application contains all the sub-projects. This means that the interfaces between the sub-systems lie within the entire application boundary. The project function point count is the sum of the number of function points of the sub-projects and can be higher than the number of function points of the entire application (application function point count), because in this situation, an internal logical file of a particular sub-project (for example) is also counted as an internal logical file or as an external interface file for another sub-project that makes use of the same internal logical file.
- A practical way of figuring out whether a certain function lies within a boundary of an application requested by a user is to ask whether the user really wants to pay for this function.
- If in doubt, consult with the user if possible.

4.6.2 The project function point count of development projects

A development project realizes an entirely new application. When a development project is split up into a number of sub-projects, then each sub-project must be treated as an independent development project when determining a *project function point count*.

The steps to be taken here are as follows:

Step 1: Determine the number of function points of each (sub)system to be realized.

Step 2: Count the number of function points of the conversion software.

These function points contribute to the project function point count only. The conversion software needed does not yield any additional functionality, but is only a one-time tool and therefore not a part of the application to be implemented.

See the practical situation described in section 11.14.

Step 3: Determine the number of function points of the changes in other applications that are being realized in the project. (See steps 1 through 5 in section 4.6.3.)

These function points contribute to the *project* function point count. (The new *application* function point count of the applications affected by the project must also be determined per application.)

Step 4: The size of the project is the sum of the number of function points recorded as a result of steps 1, 2, and 3.

Note: A project function point count is used often to budget. When different applications are being modified/enhanced, make certain that budgeting is done per application because, for example, there may be different development environments and, therefore, different productivity rates.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.6.3 The project function point count of enhancement projects[2]

An enhancement project considers enhancements to one or more existing applications. This means that functionality can be added to, changed in, and deleted from these applications.

The steps required to determine the project function point count in function points are as follows:

Step 1: Determine which transactions and/or logical files are going to be deleted from the existing application(s) and determine how many function points they represent (*DEL*).

Step 2: Establish which transactions and/or logical files change. Then determine the number of function points they represent after the change (*CHGA*).

Step 3: Identify which transactions and/or logical files are going to be added to the application(s) and establish how many function points they represent (*ADD*).

Step 4: Calculate the project function point count for the enhancement project as follows (*EFP*):

$$EFP = DEL + CHGA + ADD$$

Step 5: If conversion software has to be made as a result of changes, determine the number of function points it entails and add it to the project function point count determined in step 4.

Note: Existing internal logical files and external interface files that are used by the functions to be added, changed, or deleted, but are themselves not changed during an enhancement project, are *not* counted as internal logical files or as external interface files when the project function point count is being determined.

4.6.4 The project function point count during the replacement of an application

It is often necessary to replace applications that have been operational for a longer period of time with applications that are more efficient and that meet the requirements of current-day information technology. This is done with re-engineering projects.

Because an entire application has to be built in such a case, determining the project function point count occurs in the same fashion as when dealing with development projects. (See section 4.6.2.)

4.7 FPA in specific situations

You can apply FPA in different situations, but must deal with it in a specific way for each particular situation. This section will explain the use of FPA in the following situations:

- Counting on the basis of traditional design
- Counting application packages
- Counting from screens
- Counting during prototyping
- Counting in GUI environments

---

[2] The NESMA has published a manual “FPA for software enhancement” that goes extensively into the use of FPA during enhancement. The method for counting enhancement projects described in this International Standard is a further refinement of the method discussed here.

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.7.1 Counting on the basis of traditional design

Via models, a traditional design describes the functionality a user desires. Generally speaking, a data model and a process model are encountered here. A data model can take the shape of an Entity Relationship Diagram to which a description of entity types, attribute types, and relationships has been linked. A process model on the other hand can take the shape of a Data Flow Diagram to which a description of the functions and the data flows is linked. When counting is done from a traditional design, the models cited serve as the basis for the FPA, as well as the screen layouts and list layouts often encountered. (The screen and list layouts are not necessary, but are very useful.) Data functions are derived from the data model, and the transactional functions that must be identified are determined from the process model.

The major advantages of carrying out a function point analysis from a traditional design are that:

- counting is done from the perspective of the functionality requested and defined, in which technological considerations hardly play a role
- the design is a complete illustration of the functionality requested

4.7.2 Counting application packages

A package implemented in an organization represents a certain quantity of functionality.

The function point count carried out during the phase in which the specifications are recorded is a reflection of the amount of functionality a user desires. This count is independent of the solution to be chosen and is separate from the implementation or non-implementation of a package.

At the close of the specification phase, a decision is made whether to build an application or to buy a package that satisfies the functional requirements.

The way in which counting should be carried out depends a lot on a package's available documentation. Other than that, the approach here is no different than the one given in section 4.1. Next we will discuss the process in which the data functions (logical files) and transactional functions (transactions) of a package can be identified.

General

When an application package is considered a possible solution, the first step to be taken is to determine whether a suitable application package is available that can run on the technical infrastructure desired. If so, the following about the chosen package must be determined:

1. What functionality desired by the user can the package provide, and what is the number of the function points of this functionality?
2. What functionality desired by the user can the package *not* provide, and what is the number of function points of this functionality?
3. What functionality not desired by the user does the application package provide, and what is the number of function points of this functionality?

Group 1 represents the function points counted during the specification phase that a package can supply.

Group 2 makes up the number of desired function points either not provided or that must be modified. These function points should not be included in the useful application function point count of the package, because the application function point count should reflect the *desired* functionality that is provided. If a decision is subsequently made to upgrade the package so that it complies with all of the user's original requirements, the upgrade should be treated as an enhancement. (See sections 4.5.3 and 4.6.3.)

Group 3, the surplus functionality that an application package provides, can of course be a reason to choose a certain package. The additional functionality, however, falls outside the scope of the original project while it must still be paid for nonetheless. It is a part of the total application function point count of the package, but not

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

a part of the application function point count effective and useful to the user. It is not necessary to count these surplus function points.

The big advantage of using FPA when acquiring packages is that attention is focused on the relationship between the functionality provided and the cost factors that play a role in the decision either to produce your own application or to buy a package. With the help of FPA, a decision between creating and buying is made possible on the basis of functionality, costs, and the time frame within which functionality becomes available.

Counting the size of packages therefore has three possible objectives:

- To establish the price-performance ratio of a package. In other words, what will the package cost per function point? When this is assessed, only the function points of the functionality that the user or customer requires and that the package is going to provide are relevant.

- To establish the ratio between the functionality provided by a package that a user or customer requires and the total functionality the user or customer requires. This would include both additions and changes.

- To establish the ratio between the functionality provided by a package that a user or customer requires and the total functionality the package provides.

Determining the data functions (logical files) of a package

When an application package is acquired, a conceptual data model does not usually make up part of the documentation supplied. In some cases, you will have a data model of the package in third normal-form at your disposal. The data functions can be derived from this data model. In such a case, use the guidelines found in sections 5.20 and 5.21 and in chapters 5 and 6.

If neither a conceptual data model nor a data model in third normal-form is present, establish which logical files can probably be identified, using the functionality provided. You can also derive this from the physical database structure.

Determine the transactional functions (transactions) of a package

The transactional functions provided must be determined as well as possible from the functional specifications or from a user's manual if they are available.

If the documentation issued is inadequate and a test or demonstration implementation of the package is available, the transactional functions can be determined from the screens. (See section 4.7.3 for more about this.)

If only the menu structure is available, try to derive the functions from it. Choose three external inputs (add, change, and delete) when confronted with a menu option such as *maintain*. For a menu option such as *display*, choose one external inquiry and one external output.

Determine the functionality required

When determining the functionality required, you should assume the functional requirements the user or customer has imposed on a package. In other words, only those parts of the package are valued that the user has specified beforehand.

In short, use the functional requirements of the user or the customer in order to count.

If the functional requirements have not been defined, they must still be established in cooperation with the user by figuring out which logical files and which functions of the package are relevant. (Consult the above in this section for how data functions and transactional functions can be determined.)

Functionality provided by the package but not specifically requested by the user *can* be included when determining the total application function point count of the package; however, it must *not* be included when

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

determining the effective application function point count (the application function point count useful to the user).

4.7.3 Counting from screens

Application packages and existing applications often lack suitable documentation needed to carry out a function point analysis. If functional specifications are missing or are insufficient, it may be necessary to derive (a part of) the functions from the physical components of the application. One option is to start up the application concerned and count from the screens. This means that you will have to track down the functions by going through each branch of the menu tree up to and including the input and output screens. These input and output screens reflect the functionality that an application provides. A user's International Standard can also be a useful means to establish the functionality supplied.

Pay attention to the following points when you find functions in this fashion:

- Prevent double counting: The same transactional function can appear at several places in a menu structure.
- Pay attention to continuation screens: Screens inextricably bound together usually define one transactional function only.
- If a screen has continuation screens that are not bound to it inextricably, the continuation screens usually result in new transactional functions, unless the new transactional functions found in this fashion have been counted somewhere else already.
- Derive from the screens and the menu structure which reports are created by the application. Count each individual report as one external output.
- Sometimes a report with the same layout can be shown via different media; e.g., display screen and printer. When the logical processing is the same here, only one external output should be counted (see section 9.1).
- An external inquiry may be counted only if it is cited explicitly as such in the menu structure. Displaying data occurs rather often in practice as part of an external input, namely when data is changed and/or deleted. Given such a case, the displaying of data is not counted as an external inquiry.
- Count list functions as indicated in section 5.16.
- Using the documentation or menu options, determine which transaction files exist. Count these files accordingly as either external inputs or as external outputs. Also determine how many external inputs and external outputs should be identified per transaction file.
- Try to derive from the maintenance functions which logical files the user can maintain; in other words, which internal logical files are present.
- Often, you will not be able to substantiate the complexity of transactional functions and data functions when counting from screens. In that case, value each transactional function as average and each data function as low.
- Count the menu structures as indicated in section 5.15.
- Try to gain insight into (batch) functions executed periodically. Sometimes you will be able to see from remarks on screens, in documentation, or in menus whether a transaction run will operate or should have operated (at night, for example).

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.7.4 Counting when prototyping

Prototyping is used:

- As a strategy in order to determine functional specifications
- As an aid to develop the organization of screens and dialogs for known functional specifications.

Both situations are discussed in further detail below.

Prototyping to determine specifications

When you use prototyping as a design strategy to determine the functional specifications of an application, you should be careful in applying FPA. Prototyping is usually applied when there is uncertainty about the information problem and about the requirements that users expect of a solution for that problem. The causes of this uncertainty are due to a communication gap between the developer and user, and to a shift or change in the information need when the user has gained experience with the application. As long as uncertainty exists about the final functionality of an application to be developed, a function point count will not provide reliable insight into the ultimate size of this application. If necessary, an indicative function point count can be carried out if a (rudimentary) data model is available. It is important, however, to document specifications when prototyping, too, so a function point count can be done at the end of the prototyping cycle.

Prototyping to design the user interface

If prototyping is used to design the user interface and the functional specifications are already recorded from the start, then normal FPA guidelines apply and can be used without risk.

4.7.5 Counting GUI environments

An application in an environment that uses a Graphical User Interface (GUI) (as in WINDOWS, OS2, and so forth) presents itself to a user very differently than in a traditional environment. Windows play an important role in a GUI environment. Several windows can appear on a screen at the same time. A user can receive information and pass it on using these windows, a mouse, and a keyboard. The windows have a standard construction and often have standard options at their disposal such as increasing, decreasing, moving, scrolling through, and selecting data. This environment is also characterized by the use of icons, scroll bars, radio buttons, check boxes, push buttons, list buttons, container windows, list boxes, clip boards, and so on.

FPA is applied as follows in a GUI environment. Generally speaking, you should look beyond the decorative packaging and deduce the actual functionality provided by the application in terms of internal logical files, external interface files, external inputs, external outputs, and external inquiries. The sequence in which windows are used in a transaction must be considered as a whole. The individual components of a window do not lead to additional FPA functions, no matter how user friendly they may appear in design. Many of the components in a traditional environment correspond to a text field and are counted accordingly as a data element type.

Functionality that the GUI environment provides, or functionality expected as a standard in the GUI environment and obtained (almost) automatically by tools in the GUI environment, can be seen as an extension of the operating system and does not lead to the counting of additional functions or data element types.

4.8 Illustration: FPA and the system life cycle

A function point count can be carried out only when a certain minimum of specifications is present. Which specifications are available and the way in which they are supplied depend on the phase within the system life cycle, on the nature of the project, and on the methods that are used.

Section 3.8 will explain this further and, in doing so, will illustrate the phasing according to a general system life cycle. Comparable remarks apply to every other arbitrary phasing method. We have left it to the reader to compare this general system life cycle with the methods his organization uses and to translate them.

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.8.1 FPA during the requirements phase

This phase assesses whether the development of a new or improved application is technologically, economically, socially, and organizationally feasible and worthwhile. It is the first step in the development of a particular application.

Within the boundaries established during information planning, the problem area is analyzed, and all system requirements are specified:

- What does the existing application look like?
- What functionality should the application provide?
- How can users work with the application?
- What requirements have been imposed on the quality of the application?
- What parts of existing, planned, or standard hardware and software can be used?
- How should the transfer from the existing situation to the desired situation take place?

This will result in:

- A model of the business activities
- The basic requirements for the application to be realized
- A specification of the environment requirements of the application to be implemented
- Requirements pertaining to the technological characteristics
- A global data model

Moment of count and type of count

The specifications that must be furnished at the end of the requirements phase in the system life cycle are not always adequate for carrying out an estimated function point count (section 4.2.2), but usually are sufficient for an *indicative function point count* (section 4.2.1).

Be aware that the size of an application estimated in the requirements phase is usually too low. This is a consequence of the high abstraction level of these specifications that can keep relevant details hidden. Experiences of FPA users have shown that the degree to which a count is underestimated is constant in an organization. Each organization should decide on a standard for itself in order to compensate for this. This compensation can be referred to as the scope creep.

Objective of the count

The objective of this count is to obtain an initial indication of the size of the application to be developed. This indication can be used to:

- Determine the resources necessary for the application to be developed
- Budget the project for the design of the application
- Fix a budget for the development department
- Evaluate quotations when subcontracting later phases of system development

Required documentation

In all cases, the documentation must contain the specifications as noted in section 4.2.1.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.8.2 FPA during the analysis phase

During the analysis phase, attention shifts from analyzing business activities to specifying the application that is to support these activities. Specifying the application requires an understanding of all aspects of the information that must be stored in the logical file(s), and of the way in which the user is going to communicate with the application. In fact, a number of aspect models are created that can be seen as a blueprint of the application to be developed.

This will result in:

- A model that indicates when and why information is required for which management and for which control decisions (systelogical model)
- A model that defines which data is required for which output and what the significance is of that data (infological model)
- A model in which the structure, form, and cohesion of the data of the application to be developed are established (datalogical model)
- A model that records the technical requirements for the application to be realized (technological model)

The methods used here and the documentation produced as a result will vary for each system development method.

Moment of count and type of count

Sufficient specifications emerge during the analysis phase that will enable you to make an *estimated function point count*.

Counting can be done as soon as the specifications required for an estimated count are available. Counting can occur at different moments during analysis, depending on the methods used. However, it usually takes place at the end of the analysis phase.

Be aware that the size of an application in the analysis phase is also usually estimated too low. This is a consequence of the still relatively high abstraction level of these specifications that can keep relevant details hidden. Experiences of FPA users have shown that the degree to which a count is underestimated is constant in an organization. Each organization should decide on a standard for itself in order to compensate for this. This compensation can be referred to as the scope creep. The scope creep in the analysis phase is lower than the scope creep in the requirements phase.

Objective of the count

The objective of the function point count at the end of analysis is to obtain a better indication of the size of the application to be developed. This estimate can be used to:

- Determine the resources necessary for the application to be developed
- Budget the project for the application to be developed
- Fix a budget for the development department
- Evaluate quotations when subcontracting later phases of system development
- Settle financial matters pertaining to the analysis phase via costing when the phase has been subcontracted by means of a contract. (A fixed price per specified function point is then used in such a case.)
- Record that more or less work has to be done than an earlier count indicated

Required documentation

In all cases, the documentation must contain the specifications noted in section 4.2.2.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

4.8.3 FPA during the functional design phase

During this phase, design specifications are documented to such a degree that they can be used as the basis to realize International Standard procedures or computer programs. Additionally, the logical data structure is determined so that it can be used as the basis to establish a technical data structure or database design.

Moment of count and type of count

During the functional design phase, all the specifications required for a *detailed function point count* become available. This detailed function point count will then be equal to the final function point count if no interim changes to the functional specifications appear during the next phases.

Just as in the analysis phase, counting function points can be done during the phase or at the end of the phase.

Objective of the count

The following can be accomplished on the basis of the detailed function point count at the end of the functional design phase:

- Budgeting the continuation of the project for realizing the application
- Estimating the effort and financial means required
- Evaluating a quotation for subcontracting the construction phase
- Settling financial matters pertaining to the functional design phase via costing when the phase has been subcontracted by means of a contract. (A fixed price per specified function point is then used in such a case.)
- Recording that more or less work has to be done than an earlier count indicated

Required documentation

In all cases, the documentation must contain the specifications noted in section 4.2.3.

4.8.4 FPA during the construction phase

The construction phase encompasses the actual building and testing of the application. When you do not build the application yourself but decide to acquire standard application software instead, this phase consists of evaluating and choosing one of the various standard packages examined.

Moment of count and type of count

Counting takes place during construction when changes are made to functional specifications. In essence, then, a step backwards is taken to the analysis phase or to the functional design phase where such specifications were drawn up. In the event of a change to the functional specifications at this stage, a so-called *interim function point count* is carried out. An interim function point count reflects the influence that a change has on the project function point count and on the application function point count.

Changes here involve small enhancements. If major changes must be implemented, a new project is usually defined.

The application ultimately developed must be counted once again at the end of the construction phase in order to determine the final amount of functionality that has been realized (the size of the application). This is a *detailed function point count*.

The detailed function point count at the end of the construction phase can also be derived from the detailed function point count after the functional design and all the interim function point counts during the application's construction phase.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Objective of the count

The first objective of an interim function point count is to record the costs that must be added to or deducted from the price agreed upon earlier, in the event of more or less work. The second objective of an interim function point count is to determine the new size of the application so that you have an indication of the influence of the changes on the operation and maintenance phase of the application.

The objective of the detailed function point count at the end of the construction phase is to record the number of function points that must be maintained. It also allows you to measure the stability of the design on the basis of any increase in function points. Using the project documentation, the final function point count, and the number of hours spent on the project, you can determine what the productivity has been.

Required documentation

The document to be provided at the end of this phase is not of importance to FPA. However, the documents from the analysis and functional design phases, in which functional changes have been made, *are* important. Everything stated in sections 4.2.2 and 4.2.3 for the estimated and detailed function point count should still be specified as it pertains to such changes.

4.8.5 FPA during the implementation phase

During the implementation phase any organizational changes that need to be carried out take place. Furthermore, operational data is converted and hardware and software are installed. Training and writing the user International Standard belong to an application's implementation phase.

Moment of count and type of count

No function point count is executed during the implementation phase.

4.8.6 FPA during the operation and maintenance phase

The application is now operating and should be managed and maintained in an adequate fashion.

Moment of count and type of count

Small functional changes usually belong to the operation and maintenance phase. After a change has been carried out, a detailed function point count must be done in order to re-establish the amount of functionality that must be managed and maintained.

In the event of major enhancements, a new project is usually started and counting takes place as indicated above.

Objective of the count

During this phase, use can be made of the detailed function point count of the operational application in order to make a link between the amount of functionality to be maintained and the maintenance effort required.

Another option worth investigating is the link between the quantity of function points of the applications installed and the costs of having these applications run on the available hardware.

Required documentation

All of the documentation produced up to now should be present. The detailed function point counts of all the applications to be installed and maintained must be included in this documentation.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

5 General counting guidelines

This chapter provides general guidelines that apply when a function point count is carried out. Each section of the chapter gives a general FPA guideline from a certain perspective. The titles of the section headings indicate the perspectives.

5.1 Counting from a logical perspective

Performing a function point count is based on an organization's business activities. A business activity can be supported by one or more functions. Conversely, one function can also support several business activities. How a particular application is or has been realized in a technological sense must not be taken into consideration. The "logical perspective", after all, is what is important. The technology used may not influence the determined number of function points of an application.

5.2 Applying the rules

Common sense can be necessary when applying the guidelines. Always record the supplements you have added to the guidelines, as well as any clarifications you may have made. Then report these to the NESMA Counting Guidelines Committee using the form on the final page of this International Standard: "Comments and suggestions from the reader".

Do not be subjective when establishing the complexity of a function type. *Do not use the opportunity the original theory (Albrecht '84) provides to adjust the complexity one step higher or lower!* The factors cited there allow too much room for individual interpretation.

5.3 Built functionality, not requested functionality

During the realization of an application, pieces of code can be copied from an existing application. As a result, more functionality can sometimes be built into an application than just the desired functionality. Additionally, developers may incorporate refinements into an application that were not requested because they think they are nicer.

This means that you should determine whether the supplied functionality is also the requested functionality. Non-requested functionality *does* count towards the size of the application *supplied*, but *not* towards determining the size of the application *requested*.

Requested functionality is the functionality initially specified and the functionality that results from later change requests.

In diagram form:

<table>
<tr><td colspan="3">Supplied functionality</td></tr>
<tr><td colspan="2">Requested functionality</td><td rowspan="2">Non-requested functionality</td></tr>
<tr><td>Beforehand/initial</td><td>Via change requests</td></tr>
</table>

5.4 Double counting

Functionality may be counted only once in an application, regardless of whether one or more users make use of it. This ensures that a particular logical file is counted only once in an application: either as an internal logical file or as an external interface file, but not as both. A function such as "Change customer" appearing several times in an application is counted once, provided that the functions are identical in all cases.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

5.5 Production of re-usable code

Sometimes software is set up in such a general way that it is re-usable in other applications. This kind of software can be used as a functional unit outside of the application too. This does not have any consequences for the number of function points of the application that the software develops. The production of re-usable code does not influence the number of function points.

5.6 Re-use of existing code

If you re-use existing code when constructing an application, you are making use of existing software to realize a specified functionality. Re-use makes it easier to produce a certain functionality. This functionality is included in the function point count in the usual way. In such a situation, it would be wise to work with an adapted productivity standard (fewer hours per function point) for those parts of an application in which full or partial re-usability is possible.

5.7 Screens and reports

Reports and screens are representations of the message traffic between an application and its user(s). As such, they form physical structures of messages exchanged between the application and its environment. If a description of the logical structure is lacking (i.e., if a description of the data element types which belong to the different screens and reports is not present), it will be necessary to deduce this description from the (physical) reports and screens.

Exercise caution here. A physical structure can consist of several logical structures, and a logical structure can consist of several physical structures.

See the practical situations illustrated in sections 11.10, 11.15, and 11.17.

5.8 Input and output records

Input and output records are representations of the communication between an application and other applications. As such, they form physical structures of messages exchanged between the application and its environment. If the logical structure is missing, it will be necessary to deduce it from the (physical) record layout.

Here, too, caution should be exercised. This kind of physical record can consist of several logical structures and a logical structure can consist of several physical records.

See the practical situation illustrated in section 11.11.

5.9 Security and authorization

Security functions, authorization functions, and log-on functions are often standard and, in principle, are available to all applications; therefore, they are not counted. However, if they must be built for the application to be counted, then they should be included in the function point count as well.

5.10 Operating systems and utilities

Operating systems and utilities are standard in almost every machine and, in principle, are available to all applications; therefore, they are not counted.

Modifying and tailoring an operating system to a specific application has repercussions on productivity. It does not add any additional functionality and should not be counted as a result.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

5.11 Report generators and query facilities

Three kinds of report generators and query facilities can be identified:

- Standard facilities provided by the development environment with which the user defines the selections and output products
- Specially made facilities included in an application with which the user defines selections and output products
- Regular external outputs and external inquiries in which selections and output products are fixed

Report generators and query facilities provided as part of the development environment are not counted when the size of a project or an application is being determined.

Report generators and query facilities built at the request of the user are counted as if they are a part of the application. Count the functions (internal logical files, external interface files, external inputs, external outputs, and external inquiries) on the basis of the message traffic with the user that is needed in order to compose output products and to save the queries defined.

Regular external outputs and external inquiries should be counted as indicated in chapters 8 and 9 - even when they have been created with the help a standard report generator or query facility!

See the practical situation illustrated in section 11.3.

5.12 Graphs

Just as reports, graphs can be considered output. The function point count here does not revolve around the technique required to make a certain graph and to show it to the user, but rather around the information in the graph that is used or shown. To determine an output's complexity, therefore, FPA must use the data element types required to produce a graph.

5.13 Help facilities

If an application has help facilities, then one external inquiry must be counted for the entire application for each kind of help facility found. Examples of help facilities include the following:

- Help information for the entire application
- Help information for screens
- Help information for fields (including list functions with fixed values). (See section 5.16.)

Consider the following: If help information can be called up at every screen, count only one external inquiry for the application as a whole. In total, there can be a maximum of as many external inquiries as there are kinds of help facilities in the application to be counted. The file in which help texts are stored is considered an FPA table (see section 5.20) if the help texts can be maintained.

The complexity of the help facilities' identified external inquiries must always be valued as *Low*.

See the practical situation illustrated in section 11.4.

5.14 Error messages and other messages

These are broken down into two types: computer system messages and function messages.

Computer system messages are generated by the operating system or other system software. These messages are not counted.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Function messages are generated by a transaction of an application and say something about the use of that transaction.

If function messages are linked to an external input, external output, or external inquiry, then an additional data element type is counted for all the messages together involved in the given function.

Function messages bearing on the use of several transactional function types or on the repeated use of the same transactional function type (e.g., a log report or an error report) are counted as output functions according to the guidelines stated in Chapter 8.

Note: When there is a separate entity type for the messages and it can be maintained by the user, consider the entity type as an FPA table. (See section 5.20.)

See the practical situation illustrated in section 11.5.

5.15 Menu structures

Menu structures should not be counted as a function. One data element type *is* counted, however, for each identified transactional function for the initiation of the transaction, regardless of the actual number of steps required in the menu structure. If the user himself can adapt the menu structure and the menu texts, then these functions and their corresponding logical files are counted according to the guidelines set out for counting logical files and external inputs.

See the practical situations illustrated in sections 11.6 and 11.12.

5.16 List functions

Showing a list from which a user can make a selection (e.g., a selection screen, pick function, pick list, list box, or pop-up function) is counted as an external output in accordance with the guidelines found in section 5.3. Showing a list is not considered an external inquiry because the size of such a list is not known beforehand. Any selection option available does not count as a separate function.

Bear in mind that when a list displays data stored in an entity type of the FPA table type (see section 5.20), a separate function is not counted because the functions for FPA tables ILF and the FPA tables EIF are determined for the group as a whole in a standard fashion. You can read more about this in section 5.20.

If the list shows data that is not in a logical file (internal logical file or external interface file) and not in an FPA table, then the function should be counted as a help function at field level. (See section 5.13.)

See the practical situation illustrated in section 11.20.

5.17 Browse and scroll functions

If an application produces output on the basis of a non-unique criterion or on the basis of "from", then it is counted as an external output. This is done when the selection is presented on an overview screen (one *line* for each item that satisfies the criterion), as well as when the user can browse through the detailed screens involved (one *screen* per item). No additional functions or data element types are counted for being able to browse or scroll through the output.

See the practical situations illustrated in sections 11.21 and 11.22.

5.18 Cleaning functions

In an application, functions can appear that are started on-line or automatically in a periodic fashion and that delete or archive old data. If these functions have been made to satisfy requirements that the user has imposed, count one external input for each cleaning function, taking the general guidelines into account for identifying and valuing external inputs. (See Chapter 8.) Count at the level of functions and do *not* count one external input per internal logical file.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

5.19 Completeness check on the function point count

Expect at least one external input, one external output, and possibly one external inquiry for each internal logical file. For every external interface file, expect at least one external output or one external inquiry. Also realize that an external interface file may also be read for validation or edit purposes only, so that no external output or external inquiry is needed. If these functions are missing, ask the user whether something has been forgotten and whether the file is really relevant to the application involved.

5.20 FPA tables

Entity types in an application with constants, text, decoding, and so on are referred to as FPA tables. FPA tables that can be maintained by the user with the help of the application to be counted are counted together as one internal logical file: the FPA tables ILF. FPA tables maintained by a different application together form one external interface file: the FPA tables EIF. If an entity type cannot be maintained, it may be a system table. FPA does not include this in its counting.

The following criteria must be used in order to determine whether an entity type seen from the FPA should be counted as an FPA table. As soon as one of the criterion has been satisfied, the entity type is an FPA table.

An entity type is an FPA table in the following cases:

1. The entity type can and must contain one and only one item of data (no more and no less), regardless of the number of data element types.

   Example: An entity type with data about a particular organization; e.g., name and address.

2. The entity type contains only data that is constant (in principle).

   Example: An entity type "chemical elements": mnemonic, atomic number, description (all data element types are constants).

   Example: The function point table for valuing function types as illustrated in section 3.9, where all data elements are constants too.

3. The entity type consists of a (possibly compound) key + one or more explanatory descriptions, provided that the explanations are similar.

   Example: Buyer data: buyer nr, abbreviated buyer name, full buyer name (Abbreviated buyer name and full buyer name are similar.)

   Example: Product data: product code, description Dutch, description English (descriptions in different languages are similar kinds of data)

4. The entity type contains boundary values, algorithms, and minimum or maximum values, provided that the key is single.

   Example: Telephone number range: range number, lowest telephone number, highest telephone number.

The following entity types are not an FPA table:

1. Entity types with amounts, rates, and (VAT) percentages, if they are not constants.

2. Entity types with several different kinds of data (except for those listed above).

   Example: Buyer data: buyer nr, buyer name, district-name (district name is a different data element type).

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

Notice that you must always determine whether an entity type makes up a logical file by itself or together with other entity types. See section 5.21.

Note: The above summary of entity types that are either an FPA table or (a part of) a logical file does not cover all possible cases. When in doubt, evaluate entity types within the context of this International Standard.

Do the following to determine the complexity of the FPA tables ILF and the FPA tables EIF:

- Count the number of different FPA tables that belongs to the group as the number of record types
- Count the number of different kinds of data elements of all the FPA tables together as the number of data element types

Additionally, one external input, one external output, and one external inquiry are always counted for the FPA tables *ILF*. No external inputs, outputs, or inquiries are counted for the FPA tables *EIF*.

Do the following to determine the complexity of the standard external input, external output, and external inquiry for the FPA tables ILF:

- Count the number of different entity types that belong to the FPA tables ILF as the number of referenced logical files
- Count the number of data elements of all the entity types that belong to the FPA tables ILF as the number of data element types

See the practical situations illustrated in sections 11.1, 11.7, 11.9, 11.20, and 11.24.

5.21 Deriving logical files (data functions) from a normalized data model

5.21.1 Introduction

In FPA, a logical file (an internal logical file or an external interface file) is a conceptual entity type. A conceptual entity type is made up of one or more entity types from a data model in third normal-form that, together, are considered one logical unit by a user.

The term "entity type" in this section refers to an entity type in a data model in third normal-form.

If you have a data model in third normal-form at your disposal, you will be able to determine the logical files (data functions) using the rules stated below.

Pay attention to the following matters when applying these guidelines:

- Logical files must sometimes be counted that are not in the normalized data model; e.g., historical files containing aggregated data. (See guideline 6.2.k.)
- Entity types can appear in the normalized data model that should never have been included in it; e.g., temporary files. Even though such entity types appear in the data model, they are not logical files.
- Always look critically at the data model and at the nature of the relationships, particularly at the cardinality and optionality.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005

5.21.2 Denormalization rules

The method employed to get from a data model in third normal-form to logical files is roughly as follows:

1. Determine which entity types in the data model are FPA tables, and so belong to the FPA tables ILF or the FPA tables EIF. FPA tables are valued in a specific way. See section 5.20 and guidelines 6.2.l and 7.2.g.

2. Determine which entity types are a "key-key entity" *without* other attributes. These represent an n:m relationship in the normalized data model and are not valued at all. The referring attribute (foreign key) is counted as a data element type for both logical files connected by this key-key entity.

3. Determine which entity types are a "key-key entity" *with* other attributes. Note that two situations can arise here as a result:

   a. The additional attributes are technical in nature (not requested by the user; e.g., a date/time stamp) and are not counted as data element types. If they are the only data element types, then the entity type should be dealt with as indicated in step 2 above.

   b. The additional attributes are functional in nature (required by the user), in which case, they should be treated as indicated in step 4.

4. Examine the remaining entity types as to whether they are a logical file on their own or whether together, with one or more related entity types, they make up a logical file. Determining factors are:

   - The nature of the relationship(s) with another entity type (cardinality and optionality)
   - The dependence or independence of the entity type's existence

   Both of these ideas are examined further below. See sections 5.21.3 and 5.21.4.

   After the nature of the relationship(s) has been determined, you can assess how the entity types involved should be considered using the table in section 5.21.5.

5.21.3 The nature of the relationship (cardinality and optionality)

Two entity types, Project and Employee for example, can be connected to each other via a relationship; e.g., "has".

The nature of the relationship determines how many employees can work on a project according to the data model (0, 1, or more) and how many projects a single employee can work on (0, 1, or more).

Suppose the business rule is that several employees can be used for a project (but at least one), and that a single employee must work on one project exactly. In such a case we say that the relationship between Project and Employee is 1:N.

If the business rule was that not all employees have to work on a project, the "1-side" of the relationship would be optional, and we would denote the relationship between Project and Employee as (1):N

In other situations, the "N-side" could also be optional: 1:(N) or (1):(N).

5.21.4 Independence or dependence of an entity type

Entity independence is the degree to which an entity type is meaningful in and of itself without the presence of other entity types. Below we show how you can determine whether an entity type is independent or not within the context of different situations that bear on optionality and cardinality.

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 24570:2005