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)

© ISO/IEC 2005

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2005

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS

© ISO/IEC 2005 – All rights reserved

Not for Resale

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

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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**

--`,,```-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
© ISO/IEC 2005 – All rights reserved
Not for Resale

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.*

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

--`,,```-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
© ISO/IEC 2005 – All rights reserved
Not for Resale

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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale
© ISO/IEC 2005 – All rights reserved

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.

--`,,,```-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

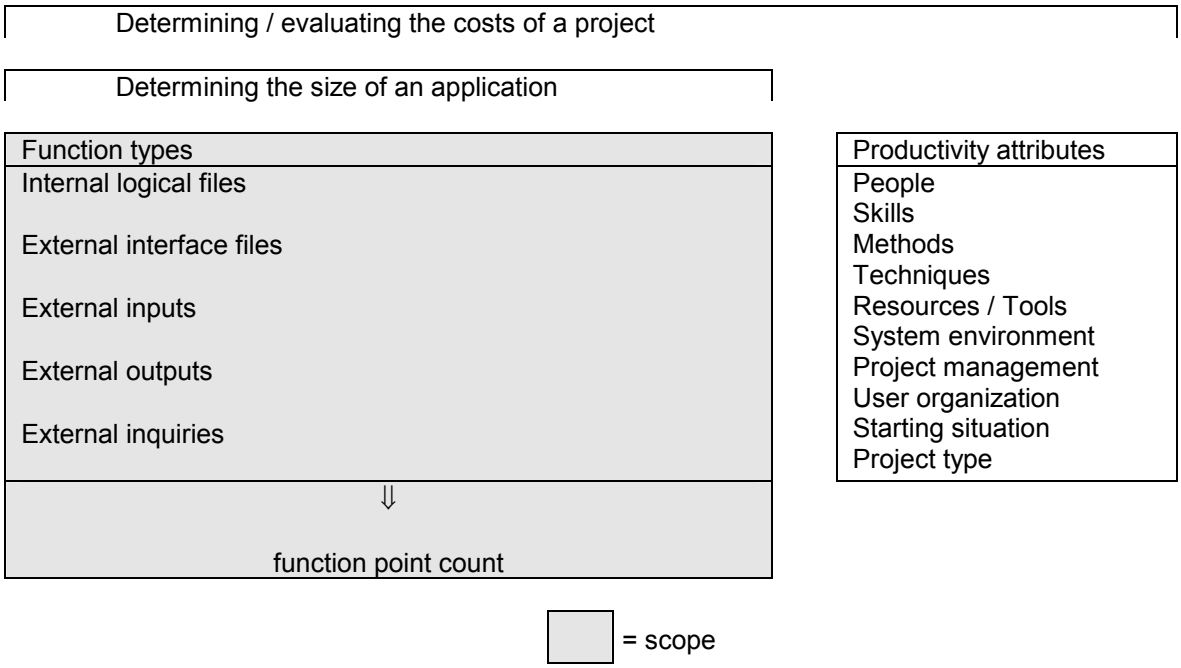


Figure 1 — Scope of the International Standard

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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*

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

--`,,`-`-`,,`,,`,`,,`---

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

		RELEASE 1	RELEASE 2	RELEASE 3
Function Points	ADD	1000	200	500
	Before CHANGE		80	220
	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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

-`,,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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).

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

--`,,`,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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).

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

--`,,`--`-`-`,,`,`,`,`--- © ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Entity independence in a (1):(N) relationship

If a relationship between two entity types (A and B) is bilaterally optional, it means that the entity types can exist independently. In such a case, FPA sees each entity type as *entity independent* in regard to each other. As the table in section 5.21.5 indicates, entity types A and B each form a logical file in FPA.

Entity dependence in a 1:N relationship

If a relationship between two entity types (A and B) is bilaterally mandatory, it means that each entity type cannot exist without its counterpart; i.e., they cannot exist independently of each other. In this case, FPA sees the entity types as *entity dependent*. As the table in section 5.21.5 indicates, entity types A and B together form one logical file in FPA.

Entity dependence or independence in a 1:(N) relationship

If a relationship between two entity types (A and B) is optional, it means that an A may exist to which no Bs are linked; e.g., as in the 1:(N) relationship between Project and Employee.

In such a situation, the idea of entity independence for entity type B plays a role in FPA. What happens to the optional side of the relationship (in this case B) when we want to delete the non-optional side of the relationship (A) when Bs are still linked to it?

Two essentially different situations are distinguished here:

(1) The deletion of A is allowed and all Bs linked to it are deleted in the same action (possibly after a message requests confirmation in which it is stated that all Bs will be deleted automatically with the deletion of A)

(2) The deletion of A is not allowed as long as Bs are still linked to it

In situation (1), the Bs are apparently not significant to the business unless they are related to an A, whereas in situation (2) they *are* significant.

Before you are allowed to delete A in the latter case, you will first have to delete all the related Bs on purpose or link these Bs to another A.

In situation (1) we say that B is *entity dependent* on A and in situation (2) that B is *entity independent* of A.

As the table in section 5.21.5 indicates, entity types A and B in FPA form one logical file in situation (1), whereas in situation (2) they each form a separate logical file.

In the example citing the relationship between Project and Employee, Employee is normally entity independent, because it is not desirable to have employee data deleted if the data of a project is deleted.

Entity dependence or independence in a (1):N relationship

The concept of entity independence as it pertains to the relationship between A and B in a (1):N relationship can also be dealt with in a similar way. These kinds of relationships, however, seldom appear in practice. The question you now have to ask yourself is, "what happens to the optional side of the relationship (in this case, A) when we want to delete the final B (the non-optional side of the relationship) linked to A?"

Two essentially different situations are distinguished here as well:

(1) The deletion of a final B linked to A is allowed in which A is also deleted in the same action (possibly after a message requests confirmation in which it is stated that A will be deleted automatically with the deletion of B)

(2) The deletion of B is not allowed as long as A is still linked to B

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

In situation (1), A is apparently not significant to the business unless it is related to one or more Bs, whereas in situation 2 it *is* significant.

Before you are allowed to delete the final B in the latter case, you will first have to delete the related A on purpose, or you must link this A to one or more other Bs.

In situation (1) we say that A is *entity dependent* on B and in situation (2) A is *entity independent* of B.

As the table in section 5.21.5 indicates, entity types A and B in FPA form one logical file in situation (1), whereas in situation (2) they each form a separate logical file.

5.21.5 Conversion table: from normalized entity types to logical files

In the table on the following page, A and B are two entity types (not an FPA table and not a key-key entity) (see section 5.21.2 point 1, 2, and 3) from the normalized data model that are connected to each other via a relationship.

Table 2

Type of relationship between A and B	How to count A and B	Condition
(1) : (N)	2 LFs	
1 : N	1 LF, 2 RETs, sum DETs	
1 : (N)	1 LF, 2 RETs, sum DETs	If B is entity dependent on A
	2 LFs	If B is entity independent of A
(1) : N	1 LF, 2 RETs, sum DETs	If A is entity dependent on B
	2 LFs	If A is entity independent of B
(1) : (1)	2 LFs	
1 : 1	1 LF, 1 RET, sum DETs	
1 : (1)	1 LF, 2 RETs, sum DETs	If B is entity dependent on A
	2 LFs	If B is entity independent of A

Legend: LF = Logical file (ILF or EIF)

RET = Record type

DET = Data element type

Note: When in doubt, choose entity independent!

Bear in mind that more than two entity types can also form a logical file sometimes. In such a case, count the total number of entity types as the number of record types. In the event of a bilateral mandatory (1:1) relationship, one logical file with one record type is always the rule.

See the practical situations illustrated in sections 11.8 and 11.9.

--`,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.22 Shared use of data

The overview below provides guidelines for identifying functions when two applications share the use of data.

Table 3

	Application A	Application B
1	**Description:** Application A has an internal logical file called Customer. The current data in it is available to application B. **Counting:** Customer is an internal logical file for application A.	Application B uses the current data from the logical file called Customer in application A. Customer is an external interface file for application B.
2	**Description:** Application A makes a copy of the logical file called Customer for application B. This copy exits application A. **Counting:** Customer is an internal logical file for application A. Creating a copy of Customer is an external output. The copy of Customer is not a separate logical file.	Application B processes the copy supplied to one or more of its own internal logical files. The data from Customer is used one time only (is consumed). Reading in and processing the copy of Customer is an external input. The copy of Customer is used once only and is apparently not a permanent file. The copy is therefore not a logical file for application B.
3	**Description:** For functional reasons, application A periodically makes an extract of the data from the logical file Customer (Customer') so that other applications can reference it. Customer' remains within the boundary of application A. Differences can occur between the current data in Customer and the data in Customer'. **Counting:** Customer is an internal logical file for application A. Customer' is also an internal logical file for application A. Creating Customer' is an external input if a separate transaction maintains it.	Application B makes use of Customer'. Customer' is an external interface file for application B.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 3 (*continued*)

	Application A	Application B
4	**Description:** Application A periodically makes an extract from the logical file Customer (Customer') that is subsequently distributed to other applications. Customer' exits application A. **Counting:** Customer is an internal logical file for application A. Creating Customer' is an external output. Customer' is not permanent data for application A and is not counted as a logical file.	Application B reads Customer' in and the file is saved into a file Customer'' without undergoing any additional processing. Reading in Customer' is an external input. Customer'' is an internal logical file for application B.
5	**Description:** Application A makes an extract of the data from the logical file Customer (Customer') that is subsequently distributed to application B. Customer' exits application A. **Counting:** Customer is an internal logical file for application A. Creating Customer' is an external output. Customer' is not permanent data for application A and is not counted as a logical file.	Application B processes the data from Customer' in order to update one or more of its internal logical files. The data is used one time only (is consumed). Reading in and processing Customer' is an external input. Customer' is now used only once and therefore does not contain any permanent data. Customer', then, is not a logical file for application B.

Note: In order to be perfectly clear, and in keeping with the above mentioned guidelines, a file created several times or stored at different physical places via the same logical processing and with the same layout, should be counted as one transaction or as one logical file.

6 Internal Logical files

This chapter will further examine the data groups that the user considers to be a single logical unit and that the application to be counted must maintain. Usually these data groups are recorded in a conceptual data model. The chapter will show which internal logical files must be identified within the context of FPA. The number of internal logical files and their complexity contribute to the function point count.

6.1 Definition of an internal logical file

An *internal logical file is a logical group of permanent data seen from the perspective of the user* that meets each of the following criteria:

- It is *used* by the application to be counted
- It is *maintained* by the application to be counted

A logical group of data seen from the perspective of the user is a group of data that an experienced user considers as a significant and useful unit or object.

An equivalent to this kind of logical group of data is an object type in data modeling.

--`,,`-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Permanent means that the file remains in existence after the application has used it so that it can be used again, unlike data in other instances that is "consumed", used once.

Used means that the data is also actually made use of in the processes of the application.

Maintained indicates that it is possible to add, change, or delete data.

6.2 Counting internal logical files

6.2.a When determining internal logical files, you should depart from a conceptual data model in which data groups (object types) have been specified in a usable, identifiable, significant, and comprehensible way for the user. You cannot simply assume a data model in third normal-form. The entity types from a normalized data model should be grouped at a conceptual level here. See the guidelines in section 5.21.

6.2.b See the practical situations illustrated in sections 11.7, 11.8, and 11.9 too.

6.2.c The maintenance of data (adding, changing, or deleting it) is of decisive significance. Only data groups used *and* maintained in the application under consideration may be counted.

See the practical situation illustrated in section 11.9.

6.2.d There are three possible reasons why a defined logical file is not maintained by the application measured:

- The file belongs to a different application. When such is the case, the file is not an internal logical file, but rather an external interface file, or an input transaction file to be counted as an external input.
- The file is an internal logical file, but no maintenance functions have been defined for it when there should have been.
- The file is a technological solution and, therefore, may not be counted as an internal logical file.

6.2.e For each internal logical file, expect at least one external input, in addition to one external output or one external inquiry.

6.2.f If an internal logical file is not accessible to a user by means of an external input, determine whether it has been identified correctly as an internal logical file.

6.2.g Files introduced for technological reasons should not be counted; e.g., work files, temporary files, interim files, sort files, print files, spool files, and so on.

See the practical situations illustrated in sections 11.13 and 11.23.

6.2.h If the user requests a restart-recovery mechanism for which (for example) a check-point data file is needed, do not include this file in your count. Additionally, do not count this file when determining the complexity of the functions.

6.2.i The presence of different user-views, access paths, and/or indexes for a file identified as an internal logical file does not mean that several internal logical files are counted for this file.

6.2.j Historical files are counted as an internal logical file only if the set of data element types of a historical file is unique in relation to other files.

See the practical situation illustrated in section 11.9.

6.2.k Stay alert for historical files. The user commonly requires them, but does not always specify them on time.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2.l Entity types with constants, text, decoding, and so on within an application are counted together once as a group in the application as one internal logical file (the FPA tables ILF), provided that each of these entity types can be maintained in the application. Additionally, one external input, one external output, and one external inquiry are counted by default for the FPA tables ILF. If one of the entity types of the kind listed above cannot be maintained in the application, then it may not be counted as part of the FPA tables ILF. See the guidelines in section 5.20.

Also see the practical situations described in sections 11.1, 11.7, 11.9, 11.20, and 11.24.

6.2.m Be alert for files in which only derived data is maintained; e.g., a running register. These files may be counted as an internal logical file only if the user has specified them explicitly as a data group. If the file has been included for technological reasons (e.g., performance), then it may not be counted.

6.2.n Sometimes it is clear from the information requirements specified by the user that a file is necessary even though it does not appear in the conceptual data model. In such a case, the file should be counted as an internal logical file if it is the result of a functional requirement and can be maintained in the application. If a file is needed for technological reasons, it may not be counted.

6.2.o The internal logical files of an existing application that remain unchanged are not counted as internal logical files or as external interface files when a *project* function point count is being determined for an enhancement project.

6.3 Determining the complexity of internal logical files

Data element types

6.3.a Only those attributes used and/or maintained in the application to be counted are counted as data element types. This means that not all attributes are counted per se.

6.3.b All the data element types of the FPA tables involved in the FPA tables ILF are counted together, in so far as they are used and/or maintained in the application to be counted.

6.3.c If an internal logical file has been converted into several entity types as a result of normalization, the number of data element types of the internal logical file corresponds to the sum of the data element types (attributes) of the normalized entity types. In order to prevent double counting, the referring attributes *in* the entity types compiled into the one internal logical file are not counted.

Record types

6.3.d The number of record types for determining the complexity of an internal logical file is:

- Equal to the number of enclosed FPA tables for the FPA tables ILF
- Equal to the number of enclosed entity types for all other internal logical files (see section 5.21)

Complexity matrix

Table 4 is used to determine the complexity level of an internal logical file:

Table 4

DET / RET	1 - 19	20 - 50	51+	
1	L	L	A	L = Low
2 - 5	L	A	H	A = Average
6+	A	H	H	H = High

© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

DET = Data element type

RET = Record type

The availability of data regarding the number of record types and the number of data element types depends on the phase of the application's life cycle. If this data is not known, an identified internal logical file should be valued as *low*.

7 External Interface Files

This chapter will further examine the data groups that the user considers to be a single logical unit and that the application to be counted uses, but which a different application maintains. Usually these data groups are recorded in a conceptual data model. The chapter will show which external interface files must be identified within the context of FPA. The number of external interface files and their complexity contribute to the function point count.

7.1 Definition of an external interface file

An external interface file is *a logical group of permanent data seen from the perspective of the user* that meets each of the following criteria:

- It is *used* by the application to be counted
- It is *not maintained by the application to be counted*
- It *is* maintained by a different application
- It is *directly available to the application to be counted*

A logical group of data seen from the perspective of the user is a group of data that an experienced user considers as a significant and useful unit or object.

An equivalent of this kind of logical group of data is an object type in data modeling.

Permanent means that the file remains in existence after the application has used it so that it can be used again, unlike data in other instances that is "consumed", used once.

Used means that the data is also actually made use of in the processes of the application.

Not maintained by the application to be counted means that it is not possible to add, change, or delete data in the application to be counted.

Directly available to the application to be counted means that the application concerned always has the current data from the logical file at its disposal, even though a different application maintains this logical file.

7.2 Counting external interface files

7.2.a When determining external interface files, you should depart from a conceptual data model in which data groups (object types) have been specified in a usable, identifiable, significant, and comprehensible way for the user. You cannot simply assume a data model in third normal-form. The entity types from a normalized data model should be grouped at a conceptual level here. See the guidelines in section 5.21.

7.2.b A file is counted as an external interface file only when it is maintained by a different application than the one to be counted, but its current data is always available to the application to be counted. See section 5.22 for a further explanation.

7.2.c If an exchange of data takes place between applications via a transaction file, the transaction file is not counted as an external interface file but as an external input and/or external output instead.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Remember, an external interface file must contain functionally permanent data that can be *used* in the application more than once, unlike a transaction file whose data is *consumed* (used only once) by an application. See section 5.22 for further explanation.

See the practical situation illustrated in section 11.11.

7.2.d Expect at least one external output and/or one external inquiry for each external interface file. Occasionally, however, an external interface file is used only to permit the edit, audit or validation of the data element types on an external input.

7.2.e An external interface file must be an internal logical file of a different application.

7.2.f The presence of different user-views, access paths, and/or indexes in a file identified as an external interface file does not mean that several external interface files are counted for this file.

7.2.g Entity types with constants, text, decoding, and so on that are referenced in the application but maintained by a different application are counted together once as a group as one external interface file (the FPA tables EIF). See the guidelines in section 5.20.

7.2.h Sometimes it is clear from the information requirements specified by the user that a file is necessary even though it does not appear in the conceptual data model. If such a file is made available by a different application and the application to be counted always has the current data available from that file, then the file must be counted as an external interface file.

7.2.i A logical file may be counted as an external interface file in an application only when it is not an internal logical file for this application; i.e., in an application that does *not* maintain the logical file.

7.2.j When determining the *application* function point count, count a logical file used communally by several sub-systems of the same application once, either as an external interface file or as an internal logical file. A logical file can be counted as an external interface file only if the boundary of the application has been crossed.

7.2.k When determining a *project* function point count, count a logical file used communally by several sub-systems of the same application either as an external interface file or as an internal logical file *for each of these sub-systems*, if these sub-systems are realized in a corresponding quantity of sub-projects carried out more or less in parallel, and these sub-systems have been constructed in such a way that they must be able to exist independently; e.g., because of a phased implementation of the application or because of functional reasons.

If the application is maintained and supported as a whole after the completion of the separate projects, this logical file is counted as indicated in guideline 7.2.j when the *application* function point count is being determined.

7.2.l The external interface files of an existing application that remain unchanged are not counted as external interface files when a *project* function point count is being determined for an enhancement project.

Table 5 provides an overview to help you distinguish between internal logical files and external interface files.

Table 5

		Count as:	
Application to be counted	Other application	In the application to be counted	In the other application
Use only	Use + maintenance	EIF	ILF
Use + maintenance	Use only	ILF	EIF
Use + maintenance	Use + maintenance	ILF	ILF
Use only	Use only	*	*

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ILF = Internal logical file

EIF = External interface file

* = The asterisk indicates that logical files cannot just be *used*. There must always be an application that is responsible for maintenance.

7.3 Determining the complexity of external interface files

Data element types

7.3.a Only those attributes used in the application to be counted are counted as data element types. This means that not all attributes are counted per se.

7.3.b All the data element types of the FPA tables involved in the FPA tables EIF are counted together in so far as they are used in the application to be counted.

7.3.c If an external interface file has been converted into several entity types as a result of normalization, the number of data element types of the external interface file corresponds to the sum of the data element types (attributes) of the normalized entity types. In order to prevent double counting, the referring attributes *in* the entity types compiled into the one external interface file are not counted.

Record types

7.3.d The number of record types for determining the complexity of an external interface file is:

- Equal to the number of enclosed FPA tables for the FPA tables EIF
- Equal to the number of enclosed entity types for all other external interface files (see section 5.21)

Complexity matrix

Table 6 is used to determine the complexity level of an external interface file:

Table 6

DET / RET	1 - 19	20 - 50	51+	
1	L	L	A	L = Low
2 - 5	L	A	H	A = Average
6+	A	H	H	H = High

DET = Data element type

RET = Record type

The availability of data regarding the number of record types and the number of data element types depends on the phase of the application's life cycle. If this data is not known, an identified external interface file should be valued as *low*.

8 External inputs

This chapter further examines user requirements pertaining to the maintenance of internal logical files and the processing of control information in the application to be counted. It shows which external inputs must be

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

identified in FPA. The degree to which external inputs contribute to a function point count depends on their quantity and their complexity.

8.1 Definition of an external input

An external input is a *unique* function recognized by the user in which *data and/or control information* is entered into an application from outside that application. It is a function that the user must see as an *elementary process*.

Data is:

> data that causes an addition, change, or deletion of data in one or more internal logical files

Control information is:

> data (e.g., a signal) that activates or deactivates one or more processes of an application or that influences the effect of a transaction

An external input should be considered *unique* when it:

- consists of a set of data element types different than all other external inputs, or
- consists of the same set of data element types, but requires a *different logical way of processing*

A *logical way of processing* is:

> a method specified by the user in order to effect a desired result. This means the following within the context of external inputs:
>
> - Maintaining and possibly referencing logical files
>
> Example: When a new order is added, the file "Product" is referenced in order to see whether the product being ordered really can be ordered.
>
> - Carrying out algorithms, calculations, and validations
>
> Example: When a new order is added, logical processing consists of validations (among other things) that should be carried out to determine whether the data entered is correct.

A logical way of processing is considered *different* when:

- other internal logical files are maintained or referenced
- the same internal logical files are maintained, but there are different algorithms, calculations, and/or validations

Different technological solutions chosen to realize the same logical processing do not mean that the logical way of processing is different.

A function is an *elementary process* when two conditions are satisfied:

- The function has an autonomous meaning to the user and fully executes one complete processing of information. In other words, it is self-contained.

For example: A user enters a new employee, including the employee's salary data and his family status. From the user's perspective, entering the employee is a single and complete process. Entering the salary information and the employee's family status is a part of this process and is not separate.

- After the function has been executed, the application is in a consistent state.

--`,,```-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

For example: A user has required that an employee's salary data must be recorded when an employee's name is being entered. Entering only the employee's name or only the salary data of the employee results in an application in an *in*consistent state.

An external input can entail both the data and control information that a user enters directly and the information that has been received from other applications.

8.2 Counting external inputs

8.2.a Count each input of data in so far as it:

- is an elementary process *and*
- has been specified by the user *and*
- is unique in the application to be measured *and*
- crosses the external boundary of the application *and* (usually)
- results in an addition, change, or deletion of data to, in, or from an internal logical file of the application

See the practical situations illustrated in sections 11.11 and 11.22.

8.2.b When an external input maintains several internal logical files (adds, changes, or deletes data), it is counted as a single external input if the user sees it as a whole and the opportunity to maintain each of the different files individually is not provided.

See the practical situation illustrated in section 11.11.

8.2.c If internal logical files can be maintained individually, then at least one external input is counted for each internal logical file.

8.2.d Count both the input of data the user enters directly and the input of data originating from other applications (e.g., in the form of an input file or a message). An external input can be activated by a user or by a different application. It can also be activated automatically; e.g., a batch function started automatically.

See the practical situation illustrated in section 11.11.

8.2.e External logical files with functionally permanent data may not be counted as external inputs, but as external interface files.

8.2.f Expect at least one external input for each internal logical file, though usually there will be several. Consult with the user when external inputs are lacking.

8.2.g An input screen on which different functions can be carried out (add, change, or delete) counts as different individual external inputs.

See the practical situation illustrated in section 11.10.

8.2.h An external input (e.g., to change data) that requests confirmation prior to the execution of the command entered is counted as one external input because it pertains to a single elementary process.

8.2.i Count two functions when data is added, changed, or deleted in two steps; e.g., in order to allow another employee to authorize input. The initial input is counted as one external input as is the authorization function. Each step is an elementary process. An external inquiry might appear here if it has been explicitly specified as such.

8.2.j A duplicate external input (i.e., when the set of data element types and the logical processing are the same) counted earlier is not counted again.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.k An external input which has been introduced exclusively for technological reasons (e.g., because of the technology used), but which a user has not requested, is not counted as an external input.

8.2.l Only those functions specified by the user count as external inputs.

8.2.m The menu structure is counted as indicated in section 5.15.

See the practical situations illustrated in sections 11.6 and 11.24.

8.2.n If data to be changed or deleted is presented as part of the change or delete function of an internal logical file, then this presentation of data is considered to make up a part of the external input and is not counted separately as an external inquiry.

See the practical situations illustrated in sections 11.10, 11.22, and 11.24.

8.2.o The same applies when data is made visible automatically on a change screen; e.g., when data in an input file is presented successively. It is not an external inquiry, but a part of the external input.

See the practical situation illustrated in section 11.10.

8.2.p If the external inquiry is specified separately, then "functionality" is being supplied to the user, in which case an external inquiry *is* counted. If retrieved data can then be changed using a change function, both an external inquiry and an external input are counted.

See the practical situation illustrated in section 11.10.

8.2.q If an external input consists of several input screens because not all the data required can be entered onto one screen, count this as one external input. If each input screen can also be used separately without having to go through a pre-arranged sequence, however, then count a separate external input for each input screen, provided that each screen can be considered an elementary process in and of itself.

8.2.r A time-delay between the input and processing of data is not a reason to identify additional external inputs.

See the practical situations illustrated in sections 11.13 and 11.23.

8.2.s Entering data to control an external input, output, or inquiry (e.g., selection data) is not counted as a separate function, but is counted as a part of the function involved.

8.2.t The processing of a transaction file (a file with temporary data) supplied by a different application results in several external inputs if different kinds of logical processing have been specified as a result of the data. In other situations, this can occur when several record types have been defined within the transaction file or when different processing codes have been defined within one record type.

See the practical situation illustrated in section 11.11.

8.2.u Count one external input when the input of data for the same logical processing takes place via different media.

8.2.v External inputs that seem to be the same from the outside, but that maintain other internal logical files, must be counted as separate external inputs, because they entail a different logical processing.

8.2.w If a header record and/or a trailer record with check data appears in a transaction file (e.g., total amount or total number of records), the processing of this/these record(s) is not counted as an individual external input. The check data, however, *is* counted as data element types (if requested by the user).

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2.x Sometimes the opportunity is given to select data that needs to be changed or deleted via a non-unique selection criterion. This means that characteristic data is shown of the items of the entity type that satisfy the selection criteria entered. The desired item of the entity type can then be selected. The act of displaying this data is seen as additional functionality. Count one external output for this. See section 5.16.

See the practical situation illustrated in section 11.22.

8.3 Determining the complexity of external inputs

Data element types

8.3.a Count all data element types (data and/or control information) that cross the boundary of the application to be counted.

8.3.b Count the way to get to a function and the way to start it (e.g., menu selection and function keys) as one data element type, regardless of the total number of keys.

See the practical situation illustrated in section 11.6.

8.3.c If several functions can be carried out on an input screen (e.g., add, change, and delete), then count the relevant number of data element types for each function.

8.3.d If several screens are used for an external input (e.g., first a screen into which selection data is entered and then a change screen), the combination of the screens together must be considered as a whole when counting data element types, because the function is a single elementary process. If, however, the selection data is not uniquely identifiable and the user must first choose the desired item from a number of items of the entity type selected, then you must count a separate external output. (See guideline 8.2.x.)

See the practical situation illustrated in section 11.22.

8.3.e If there are data element types for error messages as well as for other messages, or if there is a separate screen to display messages, then count as indicated in section 5.14.

8.3.f If additional data is to be displayed in response to an input, then the data element types displayed must be counted as a part of the external input. (An example of this kind of additional data is a function "Update customer data" in which a customer number is entered and, for the purpose of verification, the application displays the customer's name and address, after which a user can enter the rest of the data.)

8.3.g If a header record and/or a trailer record with check data appear(s) in a transaction file (e.g., total amount or total number of records), then this data is also counted as data element types (if requested by the user).

File types referenced

8.3.h The number of referenced logical files for each external input is determined by establishing the number of referenced logical files involved in the validation of the input and/or in the execution of the external input. The files can be either internal logical files or external interface files.

See the practical situation illustrated in section 11.2.

8.3.i The FPA tables ILF and the FPA tables EIF are not counted as referenced logical files when the complexity of external inputs is being determined. This also applies to files introduced for technological reasons. (See guideline 6.2.g.)

8.3.j If a process has been defined in which a logical file with permanent data is maintained on the basis of a transaction file (input), count this as one or more (in the event of several processing codes) external

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

inputs (the processing of the transaction file) with a minimum of one internal logical file (the permanent file).

8.3.k A transaction file (input or output) cannot be a "referenced" logical file. In other words, an external input which, for example, uses an input file in order to process its data, has nil referenced logical files unless, of course, internal logical files are updated. This also applies to temporary files.

8.3.l Files such as temporary files, sort files, and print files introduced for technological reasons are not counted. Determine whether these kinds of files are an alternative for an internal logical file or an external interface file. If such is the case, count these underlying logical files towards the complexity.

Complexity matrix

Table 7 is used to determine the complexity level of an external input:

Table 7

DET / FTR	1 - 4	5 - 15	16+	
0 - 1	L	L	A	L = Low
2	L	A	H	A = Average
3+	A	H	H	H = High

DET = Data element type

FTR = File type referenced

The availability of data regarding the number of data element types and file types referenced depends on the phase of the application's life cycle. If this data is not known, an identified external input should be valued as *average*.

9 External Outputs

This chapter further examines user requirements pertaining to output that an application produces. This output varies in size or requires further data processing. The chapter shows which external outputs must be identified in FPA. The degree to which external outputs contribute to a function point count depends on their quantity and their complexity.

9.1 Definition of an external output

An external output is a *unique* output recognized by the user that crosses the application boundary. It varies in size and/or *further data processing* is needed for it. It is a function that the user must see as an *elementary process*.

An external output must be considered *unique* if:

- the output product has a different logical layout than all the other output products, or
- the output product has *the same logical layout*, but requires *a different logical way of processing*, or
- any input part of the external output consists of a different set of data element types than the input part of all the other external outputs

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

An output product has *the same logical layout* if the set of data element types is the same. The following is allowed:

- Output product data element types with a different order

(Note: Grouping data element types in different ways (e.g., by means of intermediate headings) is seen as a different logical layout.)

- Data element types appearing in the output product, but without a value. The reasons for this may be that:

  (a) the data element type does *not* have a value in the logical file (as a result of which the data element type is not present optically)

  (b) the data element type *does* have a value in the logical file, but it is not relevant to the user

A *logical way of processing* is:

a method specified by the user in order to come to a desired result. This is understood to be the following within the context of external outputs:

- Referencing logical files

Example: When a list of employees is made, the logical file "Employee" is referenced.

- Providing further data processing such as algorithms, calculations, and/or validations.

Example: When reporting about all the employees in an organization, the logical way of processing contains the algorithms needed to calculate the total number of fixed employees, employees under hourly contracts, and all employees.

A logical way of processing is considered *different* when:

- other logical files are referenced
- the same logical files are referenced, but there are other algorithms, calculations, or validations

Selecting with a different *selection value* (whether or not with different kinds of *wild cards*) and selecting on the same field(s), but with a different *operator*, are *not* seen as a different logical way of processing.

Different technological solutions chosen in order to realize the same logical processing also do not mean that the logical way of processing is different.

A *selection value* is:

a selection on the same field(s), but with a different content. Do not confuse this with a selection on different fields.

A *wild card* is:

a specific sign indicating that different values can appear in a particular position. Examples of wild cards commonly used are the question mark ("?", representing an arbitrary character) and the asterisk ("*", representing a connected string of arbitrary characters).

An *operator* is:

equal to, greater than, smaller than, greater or equal to, smaller or equal to, not equal to, and so on.

Further data processing is the execution of algorithms or calculations made on data that has been retrieved from logical files before information is shown.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A function is an *elementary process* when two conditions are satisfied:

- The function has an autonomous meaning to the user and fully executes one complete processing of information. In other words, it is self-contained.

  For example: Consider a function in which all employees hired in a certain year can be printed. Entering the selection data and displaying or printing the employees selected is one complete processing from the perspective of the user.

- After the function has been executed, the application is in a consistent state.

  For example: An application for an invoicing system creates an output file containing data about the consumption of users who live in a certain area. Selecting the users who live in the area desired, searching for the consumption data, constructing the output file, and sending the file are considered a whole. From the user's perspective, the application is in a consistent state only when all the steps have been executed.

External outputs encompass both output products that go directly to the user in the form of reports and messages, as well as data flows that go to other applications via an on-line link or via an output file.

Only the domain of the output product is fixed for external outputs. The size of the output product, however, *does not have to be constant at all*, unlike external inquiries in which the size of the output product is constant.

9.2 Counting external outputs

9.2.a Count each output of data in so far as it:

- is an elementary process *and*
- is unique in the application to be measured *and*
- crosses the external boundary of the application *and*
- varies in size, or requires further data processing (see the definition in section 9.1).

See the practical situations illustrated in sections 11.13, 11.19, 11.20, 11.21, and 11.22.

9.2.b Count both output products supplied directly in the form of reports and messages to the user and output products supplied as output files and messages to other applications.

See the practical situation illustrated in section 11.14.

9.2.c Use the criteria below to distinguish between an external output and an external inquiry:

- The output of an external output may vary in size
- An external output does not always require input for the selecting of data
- The output of an external output may contain data that has come about with the help of further data processing (such as the calculation of data)

See the practical situations illustrated in sections 11.19, 11.21, and 11.22.

9.2.d The output part of an external inquiry may not be counted as a separate external output.

9.2.e Entering data for controlling an external output (e.g., entering selection criteria, the sort sequence desired, or the printer desired) is seen as a part of the external output and may not be counted as external input.

See the practical situation illustrated in section 11.12.

--`,,`,,`-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.2.f Expect at least one external output for each internal logical file. Consult the user when external outputs are lacking.

9.2.g An output product can comprise several external outputs. A single output product contains several external outputs when:

- the output product contains different logical layouts (see the definition of "logical layout" in section 9.1) and these logical layouts can be *retrieved individually*, or
- the output product contains different logical layouts (see the definition of "logical layout" in section 9.1) that have come about as a result of *individual logical processes*, but that have been combined for the sake of user friendliness.

Retrieved individually means that the user has the opportunity to control or select which parts are going to be printed.

Individual logical processes are said to be activated when the different parts report about a different object or when they come about as a result of other logical files. In this case, we talk about individual logical processes that result in one combined output product that can be retrieved with one command for the sake of user friendliness.

See the practical situations illustrated in sections 11.15 and 11.17.

9.2.h A report that has the same logical layout but that can be sorted in several ways counts as one external output, unless a different or an additional logical processing is needed for each sort sequence.

9.2.i Output can be intended directly for a user, a different application, or an external storage media. If the logical layout and the logical processing are identical, count it as one external output.

See the practical situation illustrated in section 11.12.

9.2.j It may be necessary to count a transaction file for a different application as several external outputs. This is the case, for example, when several record types appear in the file or when the output of different logical processes has been physically compiled in one external output file.

See the practical situation illustrated in section 11.14.

9.2.k Count only the external outputs that the user has asked for. Output products not requested by the user but introduced simply because of the technology are not counted; e.g., spool files.

9.2.l A file with functionally permanent data that the application to be counted maintains and that other applications use is counted as an internal logical file and not as an external output. Other applications that use the file, however, do count this data as an external interface file and, therefore, not as an external input.

9.2.m If the user requests an overview of possible error messages, it is not seen as a separate external output because the file with error messages is an FPA table. (See sections 5.14 and 5.20.)

9.2.n Messages (e.g., error messages) that say something about the execution of one function are linked to that one external input, output, or inquiry. They are not counted as individual external outputs, but are counted together as one data element type for the function concerned. (See section 5.14.)

See the practical situation illustrated in section 11.5 too.

9.2.o An output product with error messages or messages pertaining to the execution of different functions or to the repeated use of the same function is counted as one or more external outputs.

9.2.p An output product that is the logical result of maintenance to internal logical files (e.g., a transaction report or a processing report) is counted as one or more external outputs.

See the practical situation illustrated in section 11.23.

--`,,,```-`-`,,`,,`,`,,`---

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.2.q Count an output on the basis of several selection criteria as follows:

When the user has more options (i.e., an "and/or situation"), count the selections that mutually exclude each other. Each selection or combination of selections that exclude all others is counted separately.

See the practical situations illustrated in sections 11.19 and 11.22.

9.2.r If a header record and/or a trailer record with check data appear(s) in a transaction file (e.g., total amount or total number of records), the processing of this/these record(s) is not counted as an individual external output. The header record is comparable to the message header of a report and the trailer record to the summarizing totals in a report. The data element types from the header record or trailer record *are* counted (if requested by the user).

9.2.s If the user has the option to start several functions (either individually or in combination), in which the combination of the functions is more than the sum of their parts, you will have combination effects to contend with. Deal with the situation in the following way:

- A separate external output is counted for each function with a different processing that can be started separately.

- In principle, only one additional external output is counted for all possible combinations, unless there are different logical processes for certain (groups of) combinations. In such a case, an additional external output is counted for each (group of) combination(s) for which a different logical processing is needed.

See the practical situation illustrated in section 11.18.

9.2.t Showing a list from which a user can make a selection (e.g., a selection screen, pick function, pick list, or pop up function) must be counted as an external output if explicitly requested by the user, provided that the data originates from a logical file and not from an FPA table. Showing a list is not an external inquiry because the size of the list is not known beforehand. Any selection opportunity present is not counted as a separate function. See section 5.16 for a further explanation.

See the practical situation illustrated in section 11.20.

9.2.u Do not count any additional functions or data element types for being able to browse or scroll through produced output. See section 5.17 for a further explanation.

See the practical situations illustrated in sections 11.21 and 11.22.

9.2.v If a function results in output products whose contents are the same but that have been stated in a different language, then the function is counted as one external output because the output products, although different in language, nevertheless consist of the same set of data element types, and processing is the same for all output products. Multilingualism is valued via general system characteristic number 7, End-user efficiency.

9.2.w Even though there may be several output products, there can nevertheless be only one external output. There is one external output when:

a) the output products have the same logical layout (see the definition in

section 9.1)

and

b) the output products have come about as a result of the same logical way of

processing (see the definition in section 9.1).

See the practical situations illustrated in sections 11.17 and 11.24.

--`,,```-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.3 Determining the complexity of external outputs

Data element types

9.3.a All data element types (not their possible values) that appear in the output product generated by the external output are counted.

See also the practical situations illustrated in sections 11.16, 11.19, and 11.22.

9.3.b All control information (e.g., selection criteria, desired medium, desired printer, sort sequence, or time period of printing) at the level of data element type that must be entered to produce output are counted as data element types for the external output. Control information appearing on the output itself is counted double. Bear in mind that counting is done differently when the control information pertains to an external inquiry. (See section 10.3.)

See also the practical situations illustrated in sections 11.12, 11.16, 11.19, 11.22, and 11.24.

9.3.c All address data at the level of data element type that indicates for whom or for which device or medium the output is meant is counted.

See the practical situation illustrated in section 11.12.

9.3.d All process data making up a part of the output product (e.g., averages, results of calculations, subtotals, and totals) are counted as data element types.

See the practical situation illustrated in section 11.16.

9.3.e If logical files must be referenced for an external output, the data element types appearing and referenced there are not counted. Only data element types that cross the boundary of the application are included to determine the complexity.

9.3.f Standard data such as system date and page number are not counted as data element types.

Fixed data such as message headers, column descriptions, literals, and constants are also not counted as data element types.

Error messages or other messages generated by the external output are counted together as one additional data element type. (See section 5.14.)

See the practical situation illustrated in section 11.16.

9.3.g Function keys used to navigate through the output are not counted as data element types.

See the practical situations illustrated in sections 11.19 and 11.21.

9.3.h If a header record and/or a trailer record with check data appear(s) in a transaction file (e.g., total amount or total number of records), then this data is counted as data element types (if requested by the user).

File types referenced

9.3.i The number of referenced logical files for each external input is determined by establishing the number of referenced logical files for validating the input and/or for producing the output. A referenced file can either be an internal logical file or an external interface file.

9.3.j When an external output is bound inextricably to an external input, count the number of referenced logical files for the data processing as a whole and not just the number referenced for the external output itself. Consider, for example, the external output for making a report about the processing of a

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

number of transactions: this external output is bound inextricably to the external input for processing transactions.

9.3.k The FPA tables ILF and FPA tables EIF are not counted as a referenced logical file when the complexity of external outputs is being determined.

9.3.l Files such as temporary files, sort files, and print files introduced for technological reasons are not counted. Determine whether these types of files are an alternative for an internal logical file or an external interface file. If such is the case, count these underlying logical files towards the complexity.

Complexity matrix

Table 8 is used to determine the complexity level of an external output:

Table 8

DET / FTR	1 - 5	6 - 19	20 +	
0 - 1	L	L	A	L = Low
2 - 3	L	A	H	A = Average
4+	A	H	H	H = High

DET = Data element type

FTR = File type referenced

The availability of data regarding the number of data element types and file types referenced depends on the phase of the application's life cycle. If this data is not known, an identified external output should be valued as *average*.

10 External inquiries

This chapter further examines user requirements pertaining to output that an application produces. This output has been fully determined in size beforehand and does not require further data processing. The chapter shows which external inquiries must be identified in FPA. The degree to which external inquiries contribute to a function point count depends on their quantity and their complexity.

10.1 Definition of an external inquiry

An external inquiry is a *unique* input/output combination recognized by the user in which the application distributes an output fully determined in size without *further data processing*, as a result of the input. It is a function that the user must see as an *elementary process*.

An external inquiry must be considered *unique* if:

- the input part of the external inquiry consists of a different set of data element types than the input part of all other external inquiries, or
- the output product produced by the external inquiry consists of a different set of data element types than the output part of all other external inquiries, or
- the set of data element types of both the input part and the output product is the same, but the external inquiry requires *a different logical way of processing*

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A *logical way of processing* is:

a method specified by the user in order to come to a desired result. This is understood to be the following within the context of an external inquiry:

- Referencing logical files

Example: When displaying employee data, the logical file "Employee" is referenced.

- Validations

Example: When employee information is being retrieved, the application checks whether the user has been authorized to query this information.

A logical way of processing is considered *different* when:

- other logical files are referenced
- the same logical files are referenced, but there are other validations

Different technological solutions chosen in order to realize the same logical processing do not mean that the logical way of processing is different.

The data of the input part of an external inquiry must identify the desired output uniquely. Additionally, the size of the output must be fully determined in size.

Further data processing is, in this context, the execution of algorithms or calculations made on data that have been retrieved from logical files before information is shown.

A function is an *elementary process* when two conditions are satisfied:

- The function has an autonomous meaning to the user and fully executes one complete processing of information. In other words, it is self-contained.

For example: Consider a case in which data of a product is inquired about on the basis of its product number. The entering of the product number and the displaying of its corresponding product-data is one complete processing from the perspective of the user.

- After the function has been executed, the application is in a consistent state.

For example: The execution of only one of the two steps above would leave the application in an *in*consistent state, seen from the user's perspective.

10.2 Counting external inquiries

10.2.a Count each combination of input and output data as an external inquiry when the input of data leads to the direct generation of output and the input-output combination:

- is an elementary process *and*
- is unique in the application to be measured *and*
- crosses the boundary of the application

10.2.b External inquiries can pertain to inquiries originating directly from the user or from other applications.

10.2.c The distinction between an external inquiry and an external input can be explained in more depth by the following: the input part of an external inquiry coordinates a search action only and does not change the internal logical files whatsoever.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

See the practical situation illustrated in section 11.21.

10.2.d Do not confuse a query facility with an external inquiry. An external inquiry is a direct search action for specific data in which a single key is used generally. A query facility on the other hand is an organized structure of external inputs, outputs, and inquiries used in order to be able to formulate several inquiries with many keys and operations. FPA considers such an organized structure as an application that must be counted as such when it must be developed separately. External inputs, outputs, and inquiries, therefore, have to be counted in order to measure the query facility. See section 5.11.

See the practical situation illustrated in section 11.3 as well.

10.2.e An external inquiry is counted as an external inquiry only when the user has specified it as such. Therefore, an external inquiry must not be counted when the function is, for example, one part of a two-part data entry.

10.2.f The input part of an external inquiry may not be counted as an external input.

See the practical situation illustrated in section 11.21.

10.2.g The output part of an external inquiry may not be counted as an external output.

See the practical situation illustrated in section 11.21.

10.2.h An external inquiry must include the entering of data in order to control data processing; e.g., the entering of selection criteria. By definition, uniquely identifying data must always make up a part of the data entered.

See the practical situations illustrated in sections 11.19 and 11.21.

10.2.i Use the features below to distinguish between an external inquiry and an external output:

- The size of an external inquiry's output must be *completely determined and*
- The input of an external inquiry should consist of a search argument that is unique in its identification *and*
- The output of an external inquiry may not contain any data that has come about as a result of *further data processing and*
- Changes to internal logical files may not occur when an external inquiry is executed

See section 10.1 for more about *further data processing*.

10.2.j Do not count any additional functions or data element types for being able to browse or scroll through produced output. See section 5.17 for a further explanation.

See the practical situations illustrated in sections 11.21 and 11.22.

10.3 Determining the complexity of external inquiries

10.3.a Use the following method to determine the complexity level of an external inquiry:

1. Classify the input part of the external inquiry using the guidelines for determining the complexity of an external input and the complexity table for the input part found on the following page. Take only the data element types and the file types referenced (logical files) into account that are relevant to the *input part*.

2. Classify the output part of the external inquiry using the guidelines for determining the complexity of the external output and the complexity table for the output part found on the following page. Take only the data element types and the file types referenced (logical files) into account that are relevant to the *output part*.

3. The more complex of the two classifications determines the complexity of the external inquiry.

--`,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Complexity matrixes

Tables 9 and 10 are used to determine the complexity level of an external inquiry:

For the input part:

Table 9

DET / FTR	1 - 4	5 - 15	16+	
0 - 1	L	L	A	L = Low
2	L	A	H	A = Average
3+	A	H	H	H = High

DET = Data element type

FTR = File type referenced

For the output part: see next page.

For the output part:

Table 10

DET / FTR	1 - 5	6 - 19	20+	
0 - 1	L	L	A	L = Low
2 - 3	L	A	H	A = Average
4+	A	H	H	H = High

DET = Data element type

FTR = File type referenced

The availability of data regarding the number of data element types and file types referenced depends on the phase of the application's life cycle. If this data is not known, an identified external inquiry should be valued as *average*.

11 Practical Situations and their solutions

The practical situations below provide concrete examples of situations that a counter might be confronted with. They also show how the FPA counting guidelines should be applied.

The examples focus mainly on showing how the functions to be counted can be identified or recognized and, when applicable, state how data element types should be counted.

Each practical situation contains:

- A **Problem** to be solved
- A **Discussion** of the problems for the function point count
- A Solution
- **References** to the sections and counting guidelines involved

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.1 Standard authorization functions

Problem

A user is granted access to a computer system by the input of a computer system identification, a user-identification, and a password. This log-on procedure is the same for all applications that run on the system. In order to obtain access to a specific application, the user must type in the application-identification concerned and a password. Once he has done this, he is then authorized to carry out certain transactions of the application. The passwords are stored in a database table within the application. The system manager can change the passwords and indicate which transactions are permitted.

Is this log-on procedure counted or not? How are the authorization table and the maintenance functions counted for this?

Discussion

Do not count a function for the log-on procedure. The authorization table (that contains the passwords) is an FPA table and is included as a record type in the FPA tables ILF when logical files are counted. Changing the authorization table is not counted as a separate function because one external input, one external output, and one external inquiry is normally counted for the FPA tables ILF.

Solution

Consider the authorization table as an FPA table.

Do not count a function for the log-on procedure.

References

See guideline 6.2.l and sections 5.9 and 5.20.

11.2 Specific authorization functions

Problem

The file "Employee" in a time registration and planning system contains personal data and indicates whether someone is a project leader, a supervisor, or an employee. An employee can be authorized to fulfill one or several of these roles. The combination of these roles determines which transactions the user can carry out. For example, only the project leader can add activities to a project, whereas other project members are not authorized to do this.

Should the file Employee be counted when determining the complexity of the transaction "add activities"? After all, is this not a form of authorization?

Discussion

In order to be able to determine whether a user is allowed to carry out a certain transaction, the file Employee must be read. This is an internal logical file (not an FPA table) and should therefore be included in the count when determining the complexity of the transaction.

Solution

Include the file Employee as a referenced internal logical file when determining the complexity of the external input "add activities".

References

See sections 5.9 and 5.20 and guidelines 8.3.h.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.3 Report generator and query facility

Problem

An application has been developed in a 4GL environment. This 4GL also supplies an interactive query facility. The user can make whatever ad hoc queries he wants and produce his own reports with the help of this computerized tool.

Should this query facility and report generator be expressed in function points and, if so, in what way?

Discussion

The query facility and the report generator fall outside the boundaries of the system to be developed. They are not included in the count when determining the project function point count or the application function point count.

Solution

The query facility and the report generator are not included in the function point count when determining the application function point count or the project function point count of the application to be developed.

References

See section 5.11 and guideline 10.2.d.

11.4 Help functions

Problem

A number of help screens are to be installed in an application to be developed.

General information about the application can be obtained by striking the PF10 key; e.g., information about which modules exist and about the relationship between them. Specific information about a particular transaction can be retrieved by hitting the PF9 key; e.g., information about which fields must be filled in and the value range of the different fields.

The user cannot maintain these help texts.

How is this help facility counted?

Discussion

According to the guidelines, help screens are valued as external inquiries. The number of types of help information determines the number of functions. In this situation, there are two kinds of help information because the PF9 key provides help information at screen level and the PF10 key provides help information about the application.

The complexity of such external inquiries is always valued as low.

Solution

Count this help facility as two external inquiries of low complexity.

References

See section 5.13.

--`,,,```-`-`,,`,,`,`,,`---
© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.5 Error messages

Problem

A number of checks are carried out when a user enters customer data. If the user enters a customer number that already exists, the application displays the error message "customer already exists". When the user enters letters instead of numbers, the application displays the error message "customer number must consist of numbers".

Should each different error message be counted as a separate external output or as a separate external inquiry?

Discussion

The different error messages are not seen as separate functions, but as part of the external input, output, or inquiry involved.

The field where the error message is displayed must be counted as a data element type for the function. Therefore, do not count the number of different messages!

Solution

No additional functions are counted for error messages.

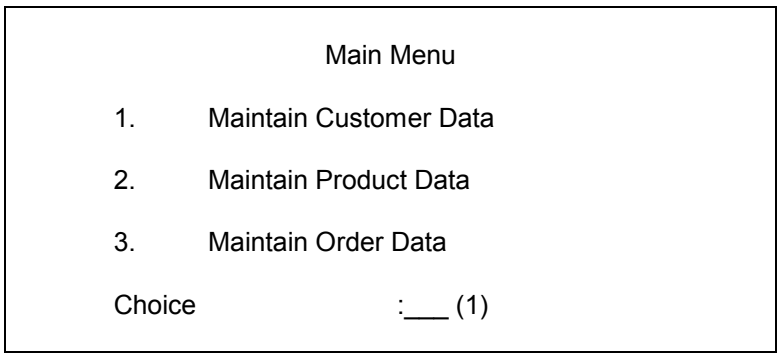
References

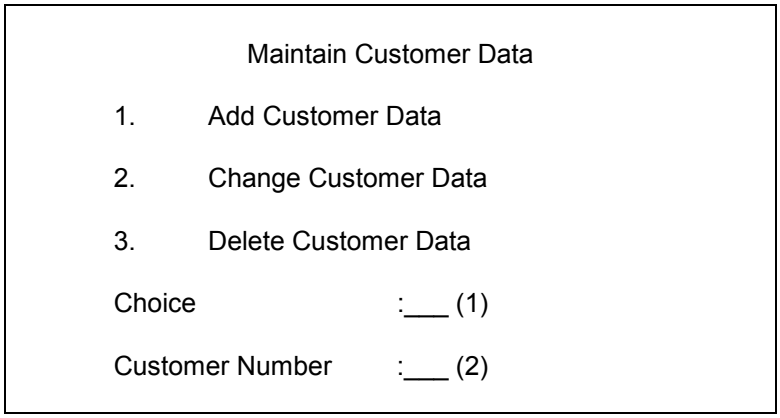
See guideline 9.2.n and section 5.14.

11.6 Menu structures

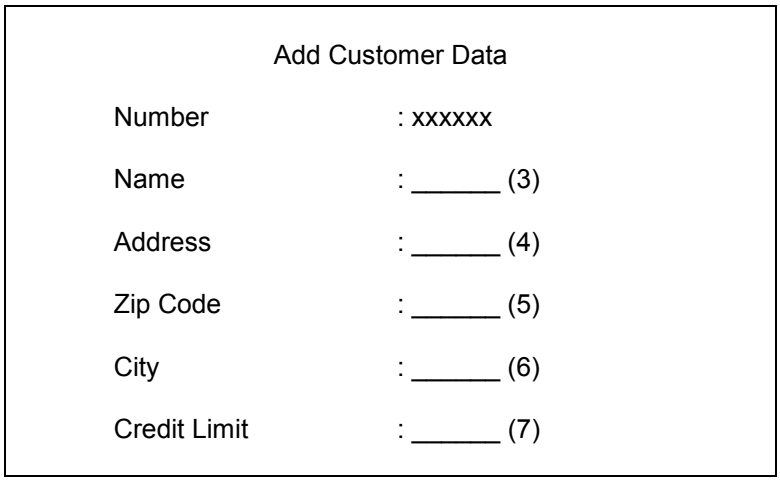
Problem

An application has the following menu structure.





© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale



How should the menu structure be counted?

Discussion

The menu structure is not counted when you are establishing the number of functions. The "Choice" field, however, is counted only as one additional data element type for the underlying functions, even though it appears twice.

Solution

Three external inputs are counted in this example: add, change, and delete customer data. The input of customer data has seven data element types as indicated by the digits in parentheses. The customer number is counted only once, just as the "Choice" field.

The menu structure is not counted when establishing the number of functions. Nevertheless, one data element type is counted for the menu structure (the "Choice" field).

References

See the guidelines 8.2.m and 8.3.b and section 5.15.

11.7 FPA tables

Problem

For a sales system that records and supports sales activities, the following entity types have been defined as part of a data model in third normal-form:

Product: product number (consists of: product group number, sequence number)

description

country of origin (code)

buyer number

price

vat code

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Country:	country code
	name of country
VAT Rate:	vat code
	vat rate
	effective date
Buyer:	buyer number
	buyer name

Functions are available for each of the entity types in order to add, change, delete, and query data. Additionally, a report with all the occurrences or specimens of data can be printed for each entity type.

Should these files be considered internal logical files? And is an FPA tables ILF or an FPA tables EIF present here? If so, what is its complexity?

Discussion

Within the framework of section 5.20, the entity type VAT Rate is not an FPA table, but an individual internal logical file. Product is also an individual internal logical file.

Because the entity types Country and Buyer are used only for decoding the codes and numbers used (i.e., they fulfill a secondary function), they should be considered an FPA table. No additional information, for example, is maintained about the buyers.

There is an FPA tables ILF because all the entity types can be maintained. Its complexity is determined as follows: The total number of entity types (two: Country and Buyer) determines the number of record types of the FPA tables ILF. The total number of data element types (four in all) of the different entity types of the FPA table type makes up the number of data element types of the FPA tables ILF. Via the complexity matrix for internal logical files, the complexity of the FPA tables ILF can be determined (*low*).

Count one external input, one external output, and one external inquiry for the FPA tables ILF, regardless of the number of entity types of which the FPA tables ILF consist.

Solution

Count three internal logical files:

- Product: consists of one record type and seven data element types. The complexity is therefore *low*.
- VAT Rate: consists of one record type and three data element types, so that the complexity is *low*.
- One internal logical file for the FPA tables. There are four data element types (country code, name of country, buyer number, buyer name) and two record types (the entity types Country and Buyer). Complexity is therefore *low*.

References

See sections 5.20 and 5.21 and guidelines 6.2.a and 6.2.l.

11.8 Denormalization

In this illustration, three examples of denormalization are given for a situation in which a 1:(N) relationship exists between two entity types. The situations 1:N, (1):N, (1):(N), 1:(1), and (1):(1) speak for themselves. (See section 5.21.3 for information about the notation method.)

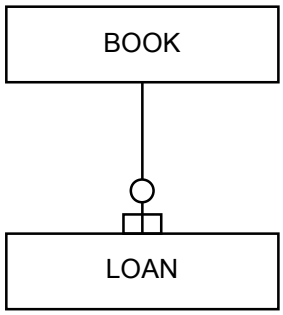
© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.8.1 1:(N) with independent existence

Problem

The normalized data model of a library application shows that there is a 1:(N) relationship between the entity types Book and Loan. A book does not have to be loaned out and, so, optionality is a factor in the relationship. If a loan is made, it always relates to one Book (no optionality).



The library's business rule is that a Book can be deleted only if a Loan is no longer linked to it.

Does this case involve one or two logical files?

Discussion

Book and Loan have a 1:(N) relationship. According to the table in section 5.21.5, the number of logical files is determined on the basis of entity dependence. Because the library may delete a Book only when a Loan is no longer linked to it, we can conclude that Loan also has a separate significance to the application apart from Book and, therefore, is entity independent from Book. (See situation 2 in the discussion about (in)dependence in a 1:(N) relationship in section 5.21.4.) There are, then, two logical files.

Solution

Count two internal logical files.

References

See section 5.21 and guideline 6.2.a.

11.8.2 1:(N) with dependent existence

Problem

The normalized data model of a library application shows that there is a 1:(N) relationship between the entity types Book and Loan. A book does not have to be loaned out and, so, optionality is a factor in the relationship. If a loan is made, it always relates to one Book (no optionality).

The business rule of this library, however, is that if Book is taken from the collection (is deleted), the library is no longer interested in Loan and, therefore, it may be deleted automatically when Book is deleted.

How many logical files must be identified in this case?

Discussion

A 1:(N) relationship exists between Book and Loan. According to the table in section 5.21.5, the number of logical files is determined on the basis of the entity dependence. Because a Book can always be deleted, and because any Loan linked to a Book may be deleted automatically with that Book, we can conclude that Loan is not significant to the application when separate from Book. Therefore, Loan is entity dependent in relation to Book. (See situation 1 in the discussion about (in)dependence in a 1:(N) relationship in section 5.21.4.) This means that there is only one logical file.

Solution

Count one logical file with two record types.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

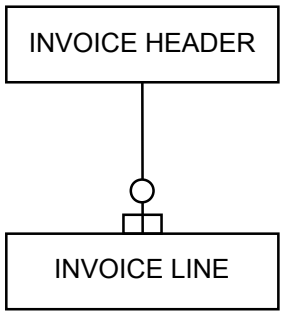
References

See section 5.21 and guideline 6.2.a.

11.8.3 1:(N) with dependent existence

Problem

The normalized data model of an invoicing system indicates that a 1:(N) relationship exists between Invoice Header and Invoice Line. The application allows users to create an Invoice Header first to which lines can be added later on; hence, the optionality. If users decide at a given moment to delete the Invoice Header, the Invoice Lines are also automatically deleted.



How many logical files should be distinguished here?

Discussion

Invoice Header and Invoice Line have a 1:(N) relationship. According to the table in section 5.21.5, the number of logical files is determined on the basis of entity dependence. Because of the business rule that any Invoice Lines linked to the Invoice Header are deleted automatically when the Header is deleted, we can conclude that Invoice Line is entity dependent in regard to Invoice Header. (See situation 1 in the discussion about (in)dependence in a 1:(N) relationship in section 5.21.4.) There is, then, one logical file called Invoice that contains the entity types Invoice Header and Invoice Line.

Solution

Count one logical file with two record types.

References

See section 5.21 and guideline 6.2.a.

11.9 Counting logical files (data functions)

Problem

Below a part of a normalized data model is illustrated.

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization All rights reserved
© ISO/IEC 2005
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

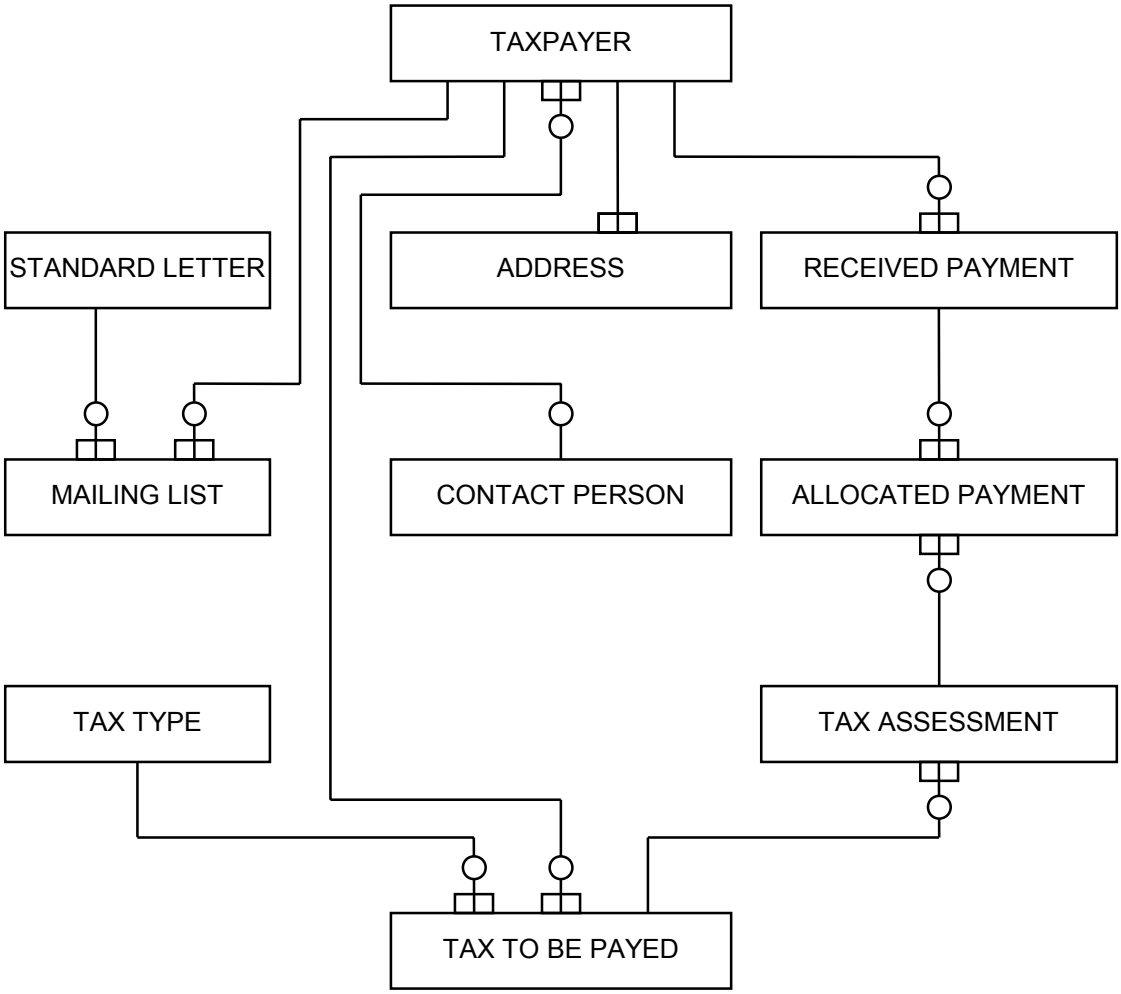


Figure 3

This data model was made on the basis of the following user specifications.

All of the entity types named are maintained by the application.

The entity type Taxpayer contains the taxpayer identification number, the name, the date of birth, and some personal information about a taxpayer.

A taxpayer can have several addresses. For example, in addition to a home address (the minimum that must be present), an invoice address and/or a post office box number may also be identified.

The entity type Standard Letter consists of a unique letter number and of a fixed text belonging to the letter.

The entity type Mailing List contains only reference keys and indicates which letter is sent to which taxpayer.

The entity type Tax Type contains the different kinds of taxes that can be charged. The composition of Tax Type is as follows: code, description, and tax amount per month. (In this particular case, the tax pertains to fixed assessments that are the same for every taxpayer.)

The entity type Tax To Be Paid records which taxes must be paid by which taxpayer. In addition to reference keys, it also contains the date on which the tax obligation becomes effective and the date on which this obligation ends (expiration date). (The latter is usually not known when the obligation becomes effective, but is recorded later.)

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The entity type Tax Assessment contains an amount, the final payment date, and the applicable tax period. An assessment always covers a fixed period: a year, half-year, quarter, or month.

The entity type Received Payment contains the amount received, the date on which the payment is received, and the amount that has still not been allocated to a Tax Assessment.

The entity type Allocated Payment contains reference keys to Tax Assessment and Received Payment, but also contains the part of the Received Payment that has been allocated for payment of the linked Tax Assessment.

The entity type Contact Person contains the names and some supplementary data about the employees of the tax department who can act as a contact person for a taxpayer. A particular contact person is assigned only when a taxpayer asks for advice. From that moment on, the taxpayer is always spoken to by the same person.

In principle, a taxpayer is entered into the system only when he is required to pay one or more kinds of tax. The taxpayer can be deleted as soon as he is no longer registered for a Tax Type (i.e., all the expiration dates in the linked entities of Tax To Be Paid have elapsed or, in other words, the taxpayer is no longer obliged to pay the tax) and no Received Payments are linked to the taxpayer anymore. When deleting the taxpayer, the entities Tax To Be Paid are deleted automatically, provided that no Tax Assessments are linked to it still.

A Tax Assessment is archived via a batch function one year after it has been paid in full. The archive file created contains the taxpayer identification number, the type of tax involved, the period the tax covers, the amount of the tax, the date on which the assessment was sent, and the date on which the assessment was paid in full. When the data is recorded in the archive, the Tax Assessment is deleted immediately together with the Allocated Payments linked to it.

A Received Payment can be deleted only if the full amount has been allocated and Allocated Payments are no longer linked to it.

Finally, a Tax Type may be deleted only if it does not have any Tax To Be Paid still linked to it.

How many logical files are present in this normalized data model? Are there any historical files?

Discussion

To analyze this data model, you should assume the denormalization rules given in section 5.21. The first question that must then be posed is whether any FPA tables are present. The description of the entity types shows that only the entity type Standard Letter meets the criteria for an FPA table. The only entity type whose status is ambiguous and can be discussed in this regard is Tax Type because it contains an amount, in addition to a code and a description. This means that it contains dissimilar kinds of data; i.e., it is not just meant for the translation of the code.

In keeping with the denormalization rules, the next question that should be asked is, "which entity types contain key-data only"? In this data model, the entity type Mailing List contains key-data only. The entity type Allocated Payment, on the other hand, contains the paid amount allocated to a Tax Assessment, in addition to reference keys; consequently, Allocated Payment does not meet this requirement. The entity type Tax To Be Paid also contains more data than just key-data.

The other eight entity types must be examined as to how many internal logical files they represent. This is done on the basis of cardinality, optionality, and entity independence. Each pair of entity types linked via a relationship is looked at to see whether they should be included in one logical file.

The relationship between Taxpayer and Contact Person is bilaterally optional. Within the context of the guidelines, then, they are independent logical files. Additionally, Contact Person does not have any relationships with other entity types and is therefore one internal logical file with one record type.

The relationship between Taxpayer and Address is a bilateral-mandatory 1:N relationship. In keeping with the denormalization rules, these two entity types belong to the same internal logical file. In order to determine

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

whether any other entity types should be included in this internal logical file, the remaining relationships of the entity type Taxpayer must be investigated.

The relationship between the entity type Taxpayer and Received Payment is a 1:(N) relationship in which Taxpayer may not be deleted as long as a Received Payment is still linked to it. This means that Received Payment is entity independent in relation to Taxpayer and does not belong to the same internal logical file as Taxpayer and Address.

The next relationship of Taxpayer that must be examined is its 1:(N) relationship to Tax To Be Paid. Here when a Taxpayer is deleted, the entities Tax To Be Paid that are linked are deleted automatically. Consequently, Tax To Be Paid is entity dependent on Taxpayer and, therefore, belongs to the same internal logical file as Taxpayer and Address. Whether any more entity types should be included in this internal logical file now also depends on the relationships of Tax To Be Paid.

The relationship between Tax To Be Paid and Tax Assessment is a 1:(N) relationship. The problem description above shows that an entity Tax To Be Paid may be deleted only if no Tax Assessment entities are linked to it anymore. Therefore, Tax Assessment has an autonomous meaning to this application and should consequently be considered entity independent in relation to Tax To Be Paid.

The relationship between Tax Type and Tax To Be Paid is also a 1:(N) relationship. A Tax Type may be deleted only if it does not have any Tax To Be Paid entities attached to it. Tax To Be Paid is therefore entity independent from Tax Type.

Now that all the relationships of Taxpayer, Address, and Tax To Be Paid have been analyzed, we can conclude that Taxpayer, Address, and Tax To Be Paid, together, make up one internal logical file with three record types.

As we have seen, Received Payment is entity independent in regard to Taxpayer. In order to determine whether this entity type is an internal logical file in and of itself, we must investigate its 1:(N) relationship with Allocated Payment. The problem description above shows that a Received Payment can be deleted only if there are no Allocated Payments attached to it anymore. This means that Allocated Payment is entity independent in regard to Received Payment. Received Payment is therefore an internal logical file with one record type.

Earlier we indicated that Tax Assessment is entity independent in relation to Tax To Be Paid. Additionally, Tax Assessment still has a 1:(N) relationship with Allocated Payment. According to the problem description above, any Allocated Payments linked to a Tax Assessment are deleted automatically when the Tax Assessment is archived and deleted. This means that an Allocated Payment is entity dependent on Tax Assessment. Tax Assessment and Allocated Payment together, therefore, make up one internal logical file with two record types.

As indicated above, Tax To Be Paid is entity independent in relation to Tax Type. Additionally, Tax Type does not have any relationships with other entity types and is not an FPA table. It is therefore an independent internal logical file with one record type.

The problem description above shows that a file with historical data does exist. This file is not included as an entity type in the data model. It is, however, required by the user. The composition of this file is different than the composition of the other internal logical files, so that a separate internal logical file with one record type must be counted for it.

--`,,`,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Solution

Count internal logical files as indicated below.

Table 11

Entity types	Count as	Number of record types
Taxpayer + Address + Tax To Be Paid	1 ILF	3
Tax Type	1 ILF	1
Received Payment	1 ILF	1
Tax Assessment + Allocated Payment	1 ILF	2
Contact person	1 ILF	1
Standard letter	Count as part of the FPA tables ILF	1
Mailing List	Not counted	
Historical Tax Assessment	1 ILF	1

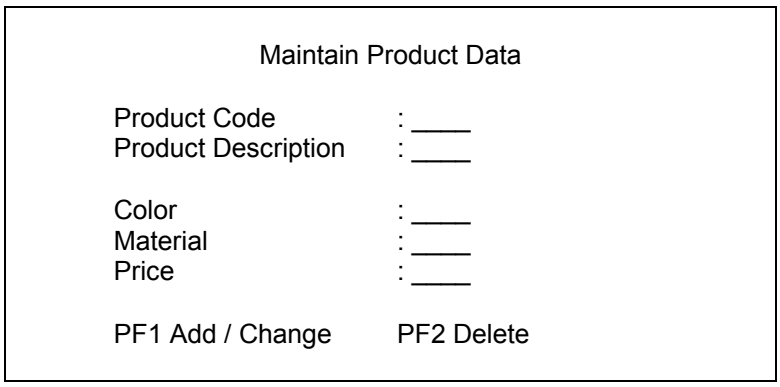
References

See sections 5.20 and 5.21 and guidelines 6.2.a, 6.2.c, 6.2.j, and 6.2.l.

11.10 Combined external inputs

Problem

An application provides the user with the option to maintain product data via the screen below.



After the user enters a product code, either an empty screen appears or a screen with product data entered earlier. When a new product code is typed in, other data can then also be entered into the remaining data fields on the screen. The data can be saved into the file by pressing the PF1 key. When a product code already used for a product is entered onto the screen, the product data can be altered and saved with PF1. A product can be deleted using PF2. When the user deletes data with this key, the application checks to see whether any stock of this product is present.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

How many and what types of functions can be distinguished here?

Discussion

Entering the data of a new product is the first external input. Do not forget that the PF1 key should be included in the count as a data element type.

Changing product data is the second external input. Note that the same set of data element types is used for another logical way of processing: to change product data. The same function key is used and the key is counted for this external input too.

Deleting product data is the third external input. From a logical standpoint, this function also differs fundamentally from the other two above. If the user considers the stock data file as an individual file, this data must be included in the count when determining the complexity of this particular external input.

Displaying product data is not counted as a separate function because the user's objective is to add, change, or delete product data. Only when the user's objective is to query the product data with this function should the displaying of data be counted as a separate external inquiry.

Solution

Count three external inputs.

References

See section 5.7 and guidelines 8.2.g, 8.2.n, 8.2.o, and 8.2.p.

11.11 Counting a transaction file

Problem

A file with shop transactions is input in a Retail Management Application. Codes distinguish one transaction from another in the application. The codes are as follows:

01 = Cash sale counter

02 = Cash return counter

03 = Sale on account counter

04 = Return on account counter

05 = Cash sale, delivery other

06 = Cash return, delivery other

07 = Sale on account, delivery other

08 = Return on account, delivery other

09 = Goods dispatched

10 = Goods received

11 = Parts retrieval by service person

12 = Parts return by service person

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13 = Old material dispatched

14 = Old material received

15 = Negative inventory difference

16 = Positive inventory difference

20 = Initial stock in store.

The following files are updated on the basis of the transaction code.

1 through 4 : Journal entry data, sales data, and stock data

5 through 8 : Journal entry data and sales data

9 through 14 : Journal entry data and stock data

15 and 16 : Inventory differences, journal entry data, and stock data

20 : Stock data

How many external inputs should be counted here?

Discussion

In this situation, particularly through the updating of different logical files, categories are made of different logical processing that can be identified. The transaction codes and what they stand for help in the categorization of the logical processing. The following external inputs are identified for processing the transactions:

1. Processing transactions that pertain to "counter activities" (transaction codes 1 through 4)

2. Processing transactions that pertain to "delivery other" (transaction codes 5 through 8)

3. Processing transactions that pertain to stock updates in the warehouse (transaction codes 9 through 14)

4. Processing transactions that pertain to inventory differences (transaction codes 15 and 16)

5. An external input for processing the initial stock (transaction code 20)

Solution

Count 5 external inputs.

References

See section 5.8 and guidelines 8.2.a, 8.2.b, 8.2.d, and 8.2.t.

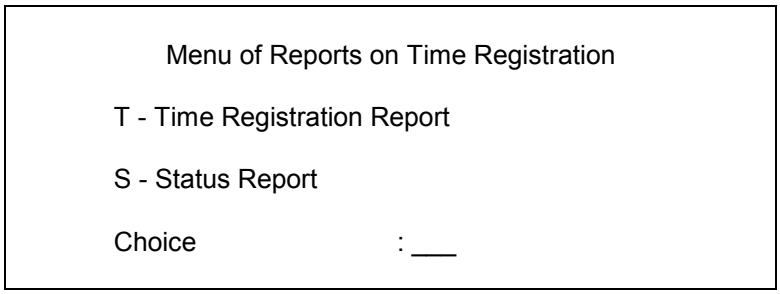
11.12 Reports on different media

Problem

Using the first selection screen on the following page, the user can make a selection from two kinds of reports (see the menu selection on the screen).

© ISO/IEC 2005 All rights reserved

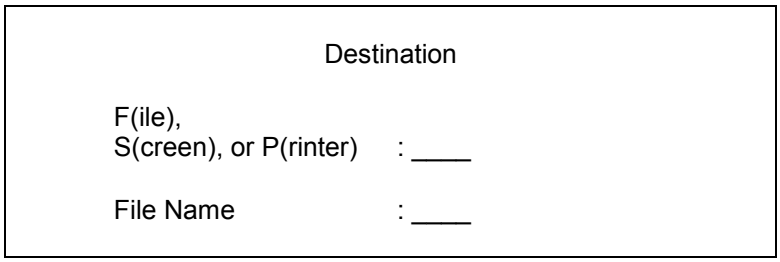
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale



Screen 1

The Time Registration Report shows time registration data that has been entered, whereas the Status Report shows a list with the status of the time registration forms.

The destination should be entered on screen 2.



Screen 2

As far as the layout and the attributes displayed are concerned, the Time Registration Report is exactly the same whether it is printed on paper, displayed on a screen, or exported to a file. The same is also true for the Status Report.

Is the report on paper a different external output than the one on screen or than the output to a file? How many external outputs are there? Is an external input counted for entering the destination?

Discussion

The criterion used to determine the number of external outputs is that each external output must be unique. An external output is unique if no other external output exists with the same logical processing and with the same set of data element types for the application concerned.

The problem description above shows that the Status Report contains different information than the Time Registration Report so that two external outputs are present.

Additionally, the layout of the Time Registration Report in this example consists of the same set of data element types, regardless of the destination. The problem description does not indicate that there is a different logical processing for the different media to which the report can be sent. The Time Registration Report is therefore counted as one external output.

This also applies to the Status Report.

The data entered for the destination is control information. This data is used only for controlling where the output is sent. This means that no external input is involved.

Solution

Count two external outputs: one for the Time Registration Report and one for the Status Report.

The choice field for the medium and the field where a file name can be filled in must be counted as data element types when complexity is being determined.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

References

See guidelines 9.2.e, 9.2.i, 9.3.b, and 9.3.c and section 5.15.

11.13 Daily and weekly processing

Problem

Each day a report of all the day's financial transactions is given. All the transactions that took place during the course of the week are placed on microfiche at the end of that week. Its layout is identical to that of the daily transaction report. After all appropriate transactions have been processed, the transaction file concerned is deleted. The content of the file is printed in the way described here only, and is ultimately placed on microfiche. The user cannot obtain access to the file in any other way.

How many external outputs must be identified? Is an external input also counted for deleting the transaction file? Does the transaction file count as an internal logical file or as an external interface file?

Discussion

The transaction file is not accessible to the user and is therefore not a logical file. Because it is a temporary file, its deletion is considered a technical matter that does not play a role when the logic of the functions is assessed. From a logical perspective, therefore, there is no difference between the daily and weekly processing. The layout of the daily report is the same as the microfiche. Because the layout and the logical processing are the same, there is only one external output.

Solution

Count one external output.

References

See guidelines 6.2.g, 8.2.r, and 9.2.a.

11.14 Conversion

Problem

The data of an existing application (PAS) was initially converted for the installation of a financial system (AIM). The PAS system is still being used and data is sent from AIM to PAS each week.

How should the conversion software and the exchange of data be counted for the AIM application?

Discussion

When software has been developed for one-time conversion (as above), this function is not included when determining the application function point count. The conversion software, however, should be counted when determining the project function point count.

The weekly transmission of data from AIM is, however, a normal external output and should be included as such in the function point count.

Solution

Count the weekly conversion as one or more external outputs of the AIM application.

Do not count the initial one-time conversion when determining the application function point count, but do count it when determining the project function point count.

--`,,```-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

References

See section 4.6.2 and guidelines 7.2.c, 9.2.b, and 9.2.j.

11.15 External outputs with summary information

Two situations are covered with in this example: one in which the summary information is not considered a separate external output and one in which it is.

11.15.1 Summary information not counted as a separate external output

Problem

The following report can be produced:

Report 1A — Overview Audio — 07/20/97
Current month: 97-07
Period : APRIL-JUNE

Country	Local Sales	Net $	QTY (x1000)	Turnover $ (x1000)	Net %	Margin (x1000)
Austria	xxx.x	xx.xx	xx.x	xxxxx	xx.x	xxxxx
Spain	xxx.x	xx.xx	xx.x	xxxxx	xx.x	xxxxx
Portugal	xxx.x	xx.xx	xx.x	xxxxx	xx.x	xxxxx
Germany	xxx.x	xx.xx	xx.x	xxxxx	xx.x	xxxxx
Europe	xxx.x	xx.xx	xx.x	xxxxx	xx.x	xxxxx
Europe	xxx.x	xx.xx	xx.x	xxxxx	xx.x	xxxxx
Asia	xxx.x	xx.xx	xx.x	xxxxx	xx.x	xxxxx
Other	xxx.x	xx.xx	xx.x	xxxxx	xx.x	xxxxx

The report consists of an unknown number of pages. The totals for Europe, Asia, and Other are printed at the bottom.

How many external outputs appear here? Should the aggregated information on the report be counted as a separate function or is it the same external output?

Discussion

There is only one external output here (the report). True enough, it consists of two sections, but the sections have the same layout. Additionally, the summarizing information for Europe, Asia, and Other are inextricably bound to the rest of the report, meaning that there is one output product whose sections are not individually retrievable. No additional logical files are accessed in order to print the information desired. The information can be derived directly from the same logical processing. The guidelines indicate that only one external output should be identified in this situation.

Solution

Count one external output.

References

See section 5.7 and guidelines 9.2.g and 9.3.d.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.15.2 Summary information counted as a separate external output

Problem

An application produces the following report:

DAILY FINANCIAL TRANSACTIONS REPORT 99-99-99 Page: 99
Date: 99-99-99

Salesperson	Transaction type	Amount
x----------x	x----------x	999.99
x----------x	x----------x	999.99
x----------x	x----------x	999.99
x----------x	x----------x	999.99

FINANCIAL TRANSACTIONS TOTALS 99-99-99 Page: 99
Date: 99-99-99

Transaction type		Number	Amount
x----------x	Day total	999	999.99
	Annual cumulative	99999	9999.99
	Daily average	999	999.99
x----------x	Day total	999	999.99
	Annual cumulative	99999	9999.99
	Daily average	999	999.99

Does one external output appear here, or are there more? Should the aggregated information be counted as a separate function on the report or is the same external output involved here?

Discussion

This report consists of one output product containing two sections. The first section is a list of all the transactions that have taken place on a given day. The second section provides daily totals, but also shows how many transactions of a certain kind have taken place in the previous year, in addition to how many per day on average. The two sections have a different layout. According to the guidelines, two external outputs should be counted if the sections can be retrieved individually *or* if they are realized via different logical processing. In this case, the sections cannot be retrieved individually. However, different logical processing is involved because data is used in the second section that is not contained in the first. The guidelines indicate that two external outputs should be identified here as a result.

Solution

Count two external outputs.

References

See section 5.7 and guideline 9.2.g.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.16 The number of data elements on a report

Problem

Report 1A — Report Audio (1) — 07/20/97
Current month: 07-97 (2)
Quarter: APRIL-JUNE (3)

	Country	QTY (x1000)	Turnover $ (x1000)	Net %	Margin (x1000)
	Austria	xx.x	xxxxx	xx.x	xxxxx
	(4)	(5)	(6)	(7)	(8)
	Spain	xx.x	xxxxx	xx.x	xxxxx
	Portugal	xx.x	xxxxx	xx.x	xxxxx
	Germany	xx.x	xxxxx	xx.x	xxxxx
	Europe	xx.x	xxxxx	xx.x	xxxxx
	(9)	(10)	(11)	(12)	(13)
	Europe	xx.x	xxxxx	xx.x	xxxxx
	Asia	xx.x	xxxxx	xx.x	xxxxx
	Other	xx.x	xxxxx	xx.x	xxxxx

This report is made by product group each quarter and contains the sales for each country. (In this case, the product group is Audio.) The report is requested via a screen. The user must enter the quarter of the report. Only those countries are printed where at least one product of a given product group has actually been sold. The totals for Europe, Asia, and Other are the sums of the respective columns. The percentage is calculated from Qty and Turnover.

How many data element types should be counted when determining the complexity of the external output?

Discussion

The data element types to be counted are denoted by the figures in parentheses. The date in the heading is standard, just as "Report 1A" in the upper left hand corner, and, consequently, is not counted. The percentage is counted once in the detailed line and once again in the total line for each geographical unit because the logical processing differs. Each column total is counted. The variable fields "Audio", "Current Month", and "Quarter" are also counted. Additionally, the data entered on the screen are counted as data element types (one in this case) when the complexity of the external output is determined. This is the so-called control information for the external output.

Solution

Fourteen data element types are distinguished in total.

References

See guidelines 9.3.a, 9.3.b, 9.3.d, and 9.3.f.

11.17 Combined external outputs

At a user's command, a commercial application prints an action list on which appear, per department, the requests for quotation that require a response or that have already received a response. The action list shows all the requests for quotations grouped by department. Each request for quotation that the application prints contains the following status: "Request under consideration", "Current quotation", "Signed contract", and

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

"Missed deal". The information displayed always covers the past week, calculated from the date of the request of the action list.

In practice, an action list can take many forms. This particular illustration presents four variants of the action list, categorized by degree of user-friendliness. Action list A, for example, provides the same information as action list D, but is not nearly as user-friendly. Action lists B and C should be considered somewhere between A and D as regards user-friendliness.

How many external outputs should be counted for each variant (A, B, C, and D)? This question should be answered within the context of two situations:

1. When the actions cannot be retrieved by status

2. When the actions can be retrieved by status, and result in one report per status.

A discussion is carried out and a solution is given for each variant (A, B, C, and D) for both situations.

11.17.1 Action list A

Action list A contains all the data elements displayed in the report below:

XXX Dept. Action List 07/20/97 10:08:32 Page: 1

Status:	Cust.	Sales-person	Req. Date	Req. for Quot.	Quot. Date	Contract Date	Expiry Date	Turnover	Reason missed
Req u/ cons.	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
Current quote	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
Signed cont.	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
Missed deal	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	xxxxxxxxxxxxxx

Variant A action list layout

Situation A.1 The actions cannot be retrieved by status

Discussion

There is one report. What must be investigated is whether there are several external outputs despite this. There can be several external outputs only when there are several sections with a different logical layout. This is not the case here and, consequently, only one external output should be counted.

Solution

Count one external output.

Situation A.2 The actions can be retrieved by status and result in one report per status

--`,,```-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Discussion

There are four reports (one for each status). The question now is whether identical functions are present. Functions are identical when:

- the logical layout of the reports is the same *and*
- the processing is the same, whereby the use of the same selection criteria with a different selection value is not seen as a different processing

In this particular situation, there are four identical logical layouts. For each report, the selection criterion "Status of the action" is used, but contains a different value in the four cases. Consequently, only one external output must be counted for the four reports.

Solution

Count one external output.

11.17.2 Action list B

XXX Dept. Action List 07/20/97 10:08:32 Page: 1

Status:	Cust.	Sales-person	Req. Date	Req. for Quot.	Quot. Date	Contract Date	Expiry Date	Turnover	Reason Missed
Req. u/ cons.	xxxxxx	xxxxxx	--/--/--	--/--/--	*)	*)	*)	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	*)	*)	*)	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	*)	*)	*)	xxxxxxxx	*)
Current quote	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	*)	*)	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	*)	*)	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	*)	*)	xxxxxxxx	*)
Signed cont.	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	--/--/--	--/--/--	xxxxxxxx	*)
Missed deal	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	*)	*)	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	*)	*)	xxxxxxxx	xxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	--/--/--	*)	*)	xxxxxxxx	xxxxxxxxxxxxxx

Variant B action list layout

Action list B is almost the same as action list A. In action list B, however, data element types that do not have a value in the internal logical file are not printed. (They cannot have a value when actions have a certain status.) The fact that a data element type cannot have a value is denoted in the report layout by *). Only a difference in appearance exists in comparison to action list A.

Discussion

Visually it seems that there are several sections with a different layout. That is, it would seem that a different logical layout can be identified for each action status. However, all distinguishable sections contain the same data element types (i.e., the column headings are the same). The only difference is that a value is not printed for certain data element types because they do not yet have a value in the specified status of the action. According to the definition of logical layout, this means that the logical layouts are the same. Counting must therefore be carried out in the same way as for action list A. This applies to both situation B.1 and situation B.2.

Solution

Count one external output for situation B.1 and one for situation B.2.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.17.3 Action list C

XXX Dept. Action List 07/20/97 10:08:32 Page: 1

Status:	Cust.	Sales-Person	Req. Date	Req. for Quot.	Quot. Date	Contract Date	Expiry Date	Turnover	Reason Missed
Req. u/ cons.	xxxxxx	xxxxxx	--/--/--	--/--/--	*)	*)	*)	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	*)	*)	*)	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	--/--/--	--/--/--	*)	*)	*)	xxxxxxxx	*)
Current quote	xxxxxx	xxxxxx	**)	**)	--/--/--	*)	*)	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	**)	**)	--/--/--	*)	*)	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	**)	**)	--/--/--	*)	*)	xxxxxxxx	*)
Signed cont.	xxxxxx	xxxxxx	**)	**)	**)	--/--/--	--/--/--	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	**)	**)	**)	--/--/--	--/--/--	xxxxxxxx	*)
.....	xxxxxx	xxxxxx	**)	**)	**)	--/--/--	--/--/--	xxxxxxxx	*)
Missed deal	xxxxxx	xxxxxx	**)	**)	**)	*)	*)	xxxxxxxx	xxxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	**)	**)	**)	*)	*)	xxxxxxxx	xxxxxxxxxxxxxxx
.....	xxxxxx	xxxxxx	**)	**)	**)	*)	*)	xxxxxxxx	xxxxxxxxxxxxxxx

Variant C action list layout

Action list C is almost identical to action list A. Just as with action list B, all action list C's data element types that do not have a value are not printed. These data element types are denoted by *). Additionally, values no longer relevant as a result of the status of the action are not printed. These fields are denoted with **).

Discussion

According to the definition, the logical layouts prove to be the same once again because all data element types appear in each section. The only difference is that a value is not printed for certain data element types because the value is no longer relevant or because it does not appear in the internal logical files. Count as you did for action list A.

Solution

Count one external output for situation C.1 and one for situation C.2.

© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.17.4 Action list D

XXX Dept. Action List 07/20/97 10:08:32 Page: 1

Requests under consideration:

Cust.	Sales-person	Req. Date	Req. for Quot.	Turnover
xxxxxx	xxxxxx	--/--/--	--/--/--	xxxxxxx
xxxxxx	xxxxxx	--/--/--	--/--/--	xxxxxxx
xxxxxx	xxxxxx	--/--/--	--/--/--	xxxxxxx

Current quotes:

Cust.	Sales-person	Req. for Quot.	Turnover
xxxxxx	xxxxxx	--/--/--	xxxxxxx
xxxxxx	xxxxxx	--/--/--	xxxxxxx
xxxxxx	xxxxxx	--/--/--	xxxxxxx

Signed contracts:

Cust.	Sales-person	Contract Date	Expiry Date	Turnover
xxxxxx	xxxxxx	--/--/--	--/--/--	xxxxxxx
xxxxxx	xxxxxx	--/--/--	--/--/--	xxxxxxx
xxxxxx	xxxxxx	--/--/--	--/--/--	xxxxxxx

Missed deals:

Cust.	Sales-person	Reason missed	Turnover
xxxxxx	xxxxxx	xxxxxxxxxxxxxxxxxx	xxxxxxx
xxxxxx	xxxxxx	xxxxxxxxxxxxxxxxxx	xxxxxxx
xxxxxx	xxxxxx	xxxxxxxxxxxxxxxxxx	xxxxxxx

Variant D action list layout

This action list contains exactly the same information as action list C, but is a bit different in layout.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Situation D.1 The actions cannot be retrieved by status

Discussion

There is one report here. What must be investigated is whether there are several external outputs despite this. There can be several external outputs only when several sections exist and each section has a different layout. In this case, four individual logical layouts would exist because each of the sections consists of a different set of data element types. (Data element types that do not have a value or that are not relevant do not appear in action list D.) This in turn would seem to indicate that four external outputs exist. According to the guidelines, however, for this to be true a user would have to be able to retrieve each section individually or there should be different logical processings for bringing about each section. The sections in this example cannot be retrieved individually. According to the counting guidelines, furthermore, different logical processing does not exist here because all the sections report about "actions" and are accomplished on the basis of the same internal logical files. One external output should therefore be counted.

Solution

Count one external output.

Situation D.2 The actions can be retrieved by status and result in one report per status

Discussion

There are four reports (one for each status). The question now is whether identical functions are involved. Identical functions are said to be present when the logical layout *and* the logical processing are the same in which the use of the same selection criterion with a different selection value is not seen as a different processing. There are four different logical layouts in this situation for the same reasons as in situation D.1. Four external outputs must then be counted as a result.

Solution

Count four external outputs.

References

See sections 5.7 and 9.1 and guideline 9.2.g.

11.18 Combination effects with functions

11.18.1 Combining one option

Problem

The user has a screen with which insurance premiums can be calculated and printed. The following options appear on the screen:

1) Premium calculation for a Man

2) Premium calculation for a Woman

The user can check off one of these options, or both simultaneously.

Suppose that the report for a Man has a different logical layout and/or logical processing than the report for a Woman. Additionally, when both options have been checked, the reports Man and Woman are printed successively and conclude with a sub-total line, a group discount, and a final-amount line.

How many external outputs should be counted in this case?

--`,,,```-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Discussion

The functions "Calculate Premium for Man" and "Calculate Premium for Woman" are unique functions and, so, one external output is counted for each. They are not external inquiries because calculations are made. An additional external output is counted because the combined report is more than the sum of its parts.

Solution

Count the following external outputs:

- Two external outputs for the unique choices Man and Woman
- One external output for the combined report

References

See guideline 9.2.s.

11.18.2 Combining several options

Problem

The user has a screen with which insurance premiums can be calculated and printed. The following options exist:

1) Premium calculation for a Man

2) Premium calculation for a Woman

3) Premium calculation for a Child

The user can check off one, two, or three of these options.

Suppose that the reports for a Man, a Woman, and a Child each have a different logical layout and/or a different logical processing. Here, again, the reports for the checked options are printed successively and conclude with a sub-total line, a group discount, and a final-amount line.

How many external outputs should be counted in this case?

Discussion

The user can now execute the following (individual or combined) functions:

Man, Woman, Child, Man + Woman, Man + Child, Woman + Child, Man + Woman + Child.

The reports for Man, Woman, and Child are each separate and unique external outputs. When combinations of the above are made, furthermore, more information becomes available than the sum of the individual parts. When combinations are made and used, however, the additional information is generated as a result of similar logical processings. (There are no different kinds of combination effects.) Consequently, one external output is counted in total for the combinations.

Note: If the processing for the combination Man + Woman + Child, for example, would have been different in comparison to other combinations, then two additional external outputs would have been counted for the combined reports. (See guideline 9.2.s.)

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Solution

The number of external outputs is:

- Three external outputs for the unique choices Man, Woman, and Child
- One external output for all of the combinations together

References

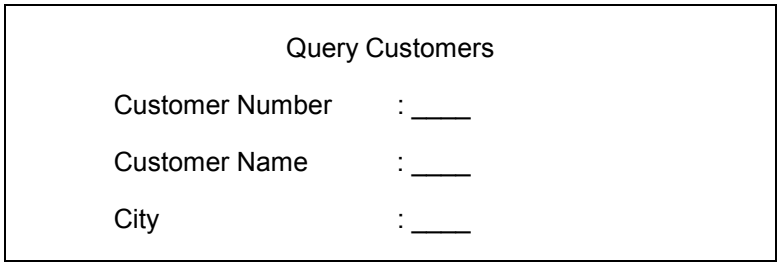
See guideline 9.2.s.

11.19 Querying with several search keys

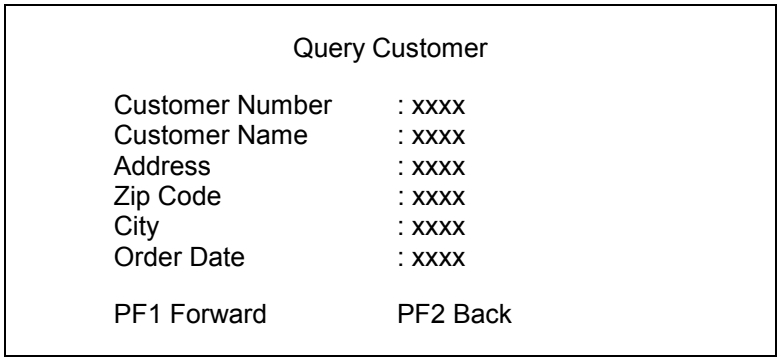
In this example, two situations are dealt with in which the option exists to retrieve data with different criteria.

11.19.1 Combination of unique and non-unique search criteria

Problem



When a unique *Customer Number* is entered into the function "Query Customers", data about the customer attached to that number will appear on the screen. The function keys (PF1 Forward and PF2 Back) are not active when this is done.



If a *Customer Name* (or part of a Customer Name) is entered, all customers with that particular name are retrieved. However, only the first customer with this name is displayed on the screen.

When *the name of a City is also entered*, only those customers that reside there are selected.

If *only the city* is entered, then all the customers from that city are selected.

The function keys allow the user to browse forward or backwards through the customers selected.

How many and what type of functions should be counted?

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Discussion

In this case, the user has the option to enter *either* the customer number *or* the customer name, and may even combine the customer name with the city. Two exclusive or separate selections are possible, each of which is considered an individual function.

Querying by customer number is an external inquiry. The size of the output is fully determined: namely, all data about a particular customer.

The input part consists of one data element type: the customer number. The output part consists of six data element types: customer number, customer name, address, zip code, city, and order date.

Querying by (a part of a) customer name and/or by city is an external output. The output varies in size because the number of customers that will be selected is not known beforehand. In this case, there is only one external output because the user has more options in which the selections he makes do not exclude each other (i.e., an and/or situation).

Eight data element types determine the function's complexity: customer number, customer name (twice), address, zip code, city (twice), and order date.

The function keys are used to navigate through the output and are therefore not counted as additional functions or data element types.

Solution

Count one external inquiry with six data element types for querying by customer number.

Count one external output with eight data element types for querying by customer name and/or by city.

References

See guidelines 9.2.a, 9.2.c, 9.2.q, 9.3.a, 9.3.b, 9.3.g, and 10.2.h.

11.19.2 Combination of non-unique search keys

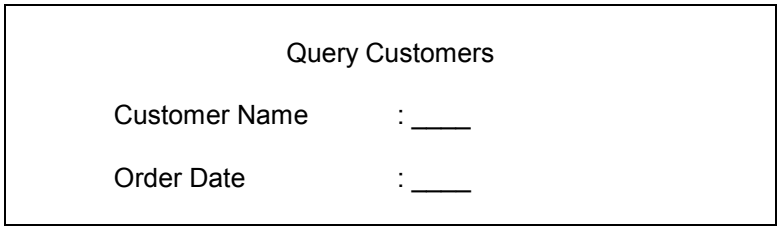
Problem

An application has the two screens below at its disposal.

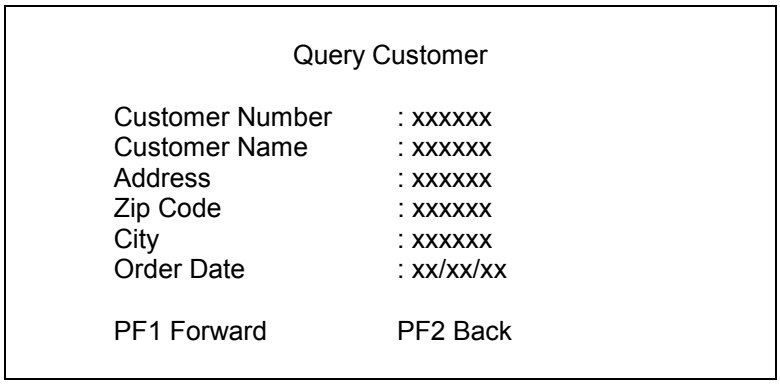
The user can query the data of a customer either via Customer Name or via Order Date.

The function keys (PF1 Forward and PF2 Back) allow the user to move to a following or a previous customer that meets the selection criterion.

How many external outputs and/or external inquiries are present here?



--`,,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale



Discussion

The output for querying by name varies in size because it is not known beforehand how many customers are selected.

The size of the output for querying by order date is also variable and cannot be predicted. The logical processing is different for both queries.

Two individual external outputs should be counted because the user must choose between querying by name or querying by order date. A combination is not an option.

The function keys are used to navigate through the output and are therefore not counted as additional functions or as data element types.

Solution

Count one external output with seven data element types for querying by name. (The seven data element types are customer number, customer name (twice), address, zip code, city, and order date.)

Likewise, count one external output with seven data element types for querying by order date.

References

See guidelines 9.2.a, 9.2.c, 9.2.q, 9.3.a, 9.3.b, and 9.3.g.

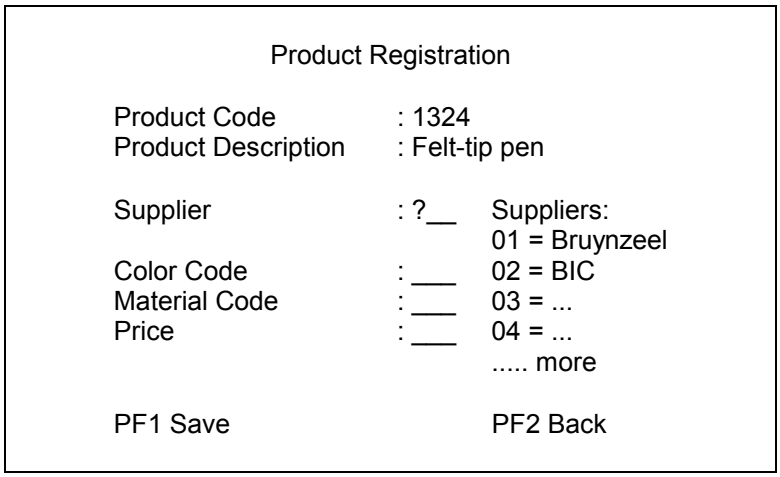
11.20 Screens with list function

Problem

When entering product data, the user can use a function key or can type in a question mark ("?") in the fields Supplier, Color Code, or Material Code in order to display all the valid codes or numbers of the field chosen, as well as the description that goes along with each of these codes or numbers (see the screen on the following page). A code or a number can then be chosen and copied from this list to the input field. The color description is retrieved from a file containing the attributes color code and color description. Material description and supplier name come from two files containing various data about the material or the supplier, respectively.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale



Should these kinds of supporting functions be counted and, if so, how?

Discussion

Count different list functions of the application in principle as separate external outputs. The list functions are external outputs because their outputs vary in size. The functions are different because (in this example) the logical layout of the list function that displays material codes is different than the list function that provides potential suppliers with their codes (i.e., the list function consists of other data element types and provides information from a different logical file).

Although this also applies to the list function with color codes, this function is not counted as a separate external output because it pertains to an FPA table. One external output is counted for the FPA tables ILF in the entire application.

Copying to the input field is not counted as a separate function.

Solution

In this example, count the list function for supplier and material each as one external output, provided that the function has not been counted elsewhere already.

Additionally, count one external input for saving product information.

Do *not* count a separate external output for the list function with color codes.

References

See guidelines 6.2.l, 9.2.a, and 9.2.t and sections 5.16 and 5.20.

11.21 Browse and scroll functions

Browse and scroll functions appear in many shapes and sizes. FPA strives to count these different shapes and sizes in the same fashion when they provide the same functionality even though they have been realized in a different way. As a result, this illustration will go into a large number of different situations and will indicate how each situation should be counted.

--`,,,```-`-`,,`,,`,`,,`---
© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.21.1 Selection via uniquely identifying data

Problem

A unique customer number is entered for the function "Show Customer Data". Once this has been done, the following situations can present themselves:

1. The data of the customer concerned is displayed. No option exists to use functions keys in order to retrieve the data of a different customer.

2. The data of the customer concerned is displayed, after which the data of the following or previous customer can be retrieved by using function keys.

3. The core data of all customers is displayed on an overview screen (one line per customer), starting from the customer number entered. The user can scroll through this data when the screen cannot display all of it because of a lack of room.

4. The core data of all customers is displayed on an overview screen (one line per customer), starting from the customer number entered. The user can scroll through this data when the screen cannot display all of it because of a lack of room. After one of the customers on this screen has been selected, the application displays its detailed data.

5. The detailed data of the customer concerned is displayed. Via a function key, a user can then request a screen-display overview of the core data of all customers (one line per customer), starting from the customer that was shown on the detailed screen. The user can then scroll through this data if the screen cannot display all of it because of a lack of room. A particular customer can then again be selected on the overview screen, after which the application displays its data on a detailed screen.

6. The core data of all customers is displayed on an overview screen (one line per customer), starting from the customer number entered. The user can scroll through this data if the screen cannot display all of it because of a lack of room. After one of the customers on this screen has been selected, the application displays its detailed data, after which the data of the following or previous customer can be retrieved by using function keys.

Which external outputs and/or external inquiries should be identified in each of the situations above?

Discussion

Situation one is clearly an external inquiry; nothing more, nothing less. The customer is determined in a unique fashion by its customer number. Only one customer has that number. No opportunity to browse is given.

Situation two also seems to be a case of an external inquiry. In reality, however, the function allows the user to browse through all the customers from a defined starting point. The entire collection of customers is provided and the quantity of customers that can appear varies. This means that one external output is present.

Situation three also has an external output. Here, too, a starting point has been defined. Several customers are displayed and the number of customers that will follow from that starting point is not known. As a result, one external output must be identified. It does not matter whether the user can scroll further with the function key because there are more customers than the screen can display. Scrolling within the same collection is not a separate function, but rather a part of the external output. The only difference between situation three and two is that all the data of a customer can be displayed in two and only core data in three.

Two functions are in fact provided in situation four. Just as in situation three, the overview screen is an external output.

Displaying data of a specific customer on the detailed screen is considered a different functionality because a different set of data element types is involved. (Only the core data of a customer appears on the overview screen, whereas all the data of a customer is displayed on the detailed screen.) Moreover, calling the function is optional. Additionally, the function itself could exist independently. Therefore, this function is also an elementary process. This, in turn, means that the displaying of detailed data is counted as a separate function. It is an external inquiry because the user cannot scroll through information once he is on the detailed screen. There is one external output and one external inquiry.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

From a functional standpoint, situation five is the same as situation four, only the screens appear in a different sequence. The sequence of screens is not important to FPA. The same functions identified in situation four are identified in five.

Just as in situation four, two functions are provided in situation six. The overview screen is once again an external output.

Displaying data of a specific customer on the detailed screen is considered a different functionality because a different set of data element types is involved. (Only the core data of a customer appears on the overview screen, whereas all the data of a customer is displayed on the detailed screen.) Moreover, calling the function is optional. The function itself could exist independently. Therefore, this function is also an elementary process. This, in turn, means that the displaying of detailed data is counted as a separate function. Unlike situation four, however, situation six *does* allow the user to browse through the detailed screens and, so, the same functionality is provided as in situation two. The displaying of detailed data is therefore counted as one external output. As a result, situation six has two external outputs in total.

Solution

Identify the following functions:

Situation 1: One external inquiry

Situation 2: One external output

Situation 3: One external output

Situation 4: One external output and one external inquiry

Situation 5: One external output and one external inquiry

Situation 6: Two external outputs

References

See section 5.17 and guidelines 9.2.a, 9.2.c, 9.2.u, 10.2.c, 10.2.f, 10.2.g, 10.2.10.2.h, and 10.2.10.2.j.

11.21.2 Selection via non-uniquely identifying data, followed by browsing

Problem

When a user enters a unique representative number for the function "Show Customer Data", the first customer of the representative concerned is displayed. Using the functions keys, the user can then browse to a previous customer of the representative or to a following one.

Is there one or more external inquiries present here and/or one or more external outputs?

Discussion

When a user enters a unique representative number, he does not know how many customers this representative has. This means that the output varies in size and that it is counted as one external output. The browse function is a part of the external output, and the function keys used to browse with are not counted as an additional function or as data element types.

Solution

Count one external output.

--`,,,```-`-`,,`,,`,`,,`---

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

References

See section 5.17 and guidelines 9.2.a, 9.2.c, 9.2.u, 9.3.g, and 10.2.j.

11.21.3 Selection via uniquely identifying data, followed by browsing after another selection

Problem

When querying customer data via a unique customer number, a user can retrieve a previous or following customer of the same representative by using function keys.

Is there one or more external inquiries present here and/or one or more external outputs?

Discussion

When a user queries the data by customer number, the output is determined uniquely by that customer number and does not vary in size. This is an external inquiry.

When function keys are used to retrieve the previous or the following customer of the same representative, the customer number of the customer displayed and the representative number are used as search keys. This means that a different logical processing is necessary. Even though the customer specifically shown has been determined uniquely, the user now browses through the collection of customers belonging to a single representative. The size of this collection varies and, therefore, an external output is present.

Solution

Count one external inquiry and one external output.

References

See section 5.17 and guidelines 9.2.a, 9.2.c, 9.2.u, 10.2.c, 10.2.f, 10.2.g, 10.2.10.2.h, and 10.2.10.2.j.

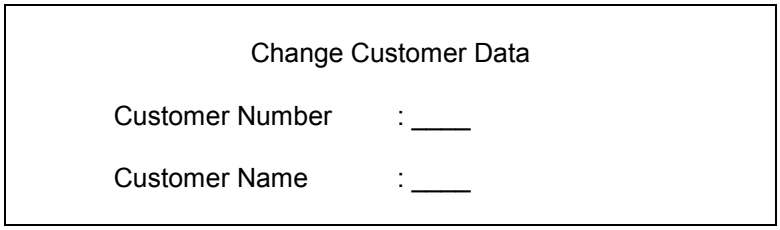
11.22 Selection screens and changing data with a search key

This example treats a change function whose objective is twofold: A user should be able to change customer data but, before he does this, he should first be able to select the customer in a user-friendly way. From a functional standpoint, this can be realized in different ways. This section will discuss two different implementations of this functionality and will indicate how counting should take place in both situations.

11.22.1 Selection via a separate selection screen

Problem

Using a menu, the user indicates that he wants to change customer data. The application subsequently presents screen 1 on which the user must enter a unique customer number or a (part of a) customer name. The user should not enter both.



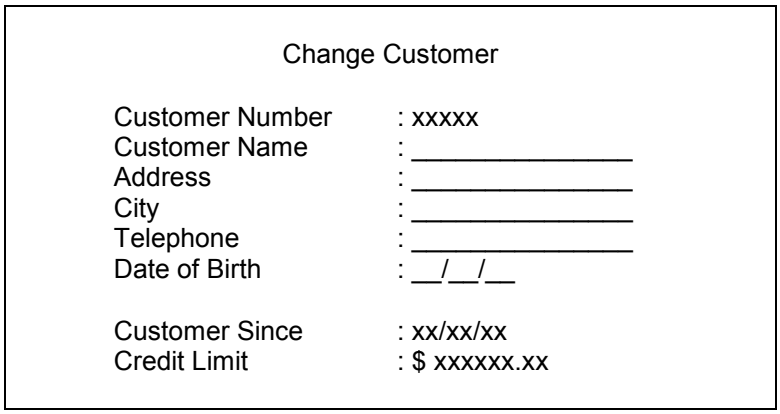
Screen 1

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

When a unique customer number is entered, the data for the customer concerned appears on the change screen. (See screen 2.)

When a customer name (or a part of a customer name) is entered, the application retrieves all customers with that name. If only one customer is found, the data of that customer appears immediately on the change screen (screen 2).



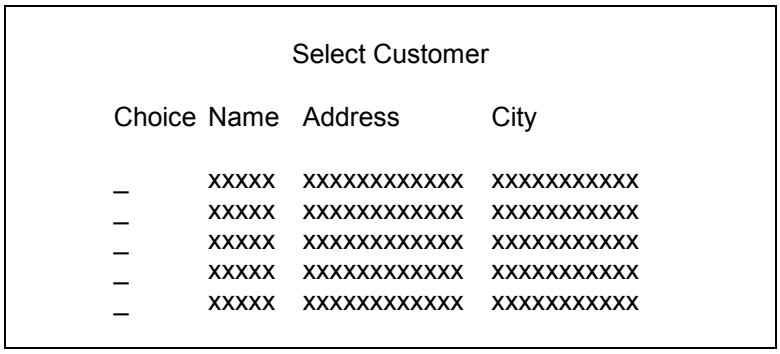
Screen 2

If several customers are found, they appear on the selection screen. (See screen 3.)

After the user enters the customer desired via the Choice field (screen 3), more extensive data of that customer appears on the change screen (screen 2).

The customer data can then be changed via screen 2.

Does the option to select data make up a part of the option to change data, or is it a separate function? Is a separate external input counted for entering the customer desired? Is the display of the customer data a separate external inquiry? How many functions should be counted in total?



Screen 3

Discussion

If the correct customer data is displayed on the change screen (screen 2) as a result of a customer number having been entered on screen 1, then the display of the data to be changed is part of the external input "Change Customer". This display is not counted as an individual external inquiry.

Counting is carried out as follows, however, when the user enters a non-unique customer name and customers meeting this criterion appear on screen 3, after which the customer to be changed can be chosen. Searching for the customer via customer name results in a displayed selection of customers on the selection screen (screen 3). This selection is not determined fully in size beforehand; i.e., the size of the selection varies

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

depending on how many customers meet the selection criterion (name). The customer desired can then be selected. This display of selected customers on a separate screen is seen as additional functionality.

Since the logical layout of this overview is different, an additional function is counted; because the output varies in size, one external output should be counted.

Sometimes an application retrieves only one customer when a user selects via customer name. When this happens, the application does not display the selection screen (screen 3). Instead, it immediately retrieves the detailed data that screen 2 displays. This is merely an optimization and is therefore considered to be part of the external output counted for the selection screen (screen 3). No additional external output or external inquiry is counted for this.

After the user has made a selection on screen 3, all the data for the customer selected is displayed on screen 2, after which changes can be made. The display of data for the customer selected on screen 2 (just as when selecting via customer number) is seen as a part of the change function and is not an individual external inquiry.

Indicating the selected customer via the Choice field does not result in a separate external input.

The change function is the same in both cases, and is therefore counted as one external input.

Solution

Count one external input for the change function.

Count one external output for displaying the customers that meet the selection criterion.

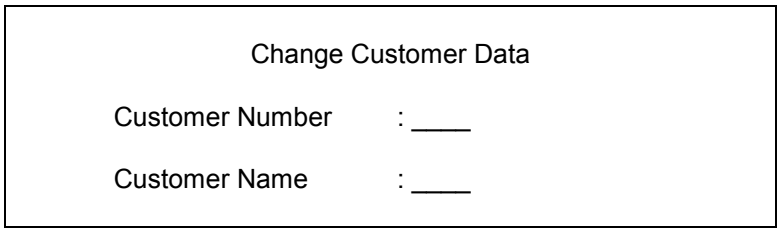
References

See sections 5.16 and 5.17 and guidelines 8.2.a, 8.2.n, 8.2.x, 8.3.d, 9.2.a, 9.2.c, 9.2.q, 9.2.u, 9.3.a, 9.3.b, and 10.2.j.

11.22.2 Selection via the change screen

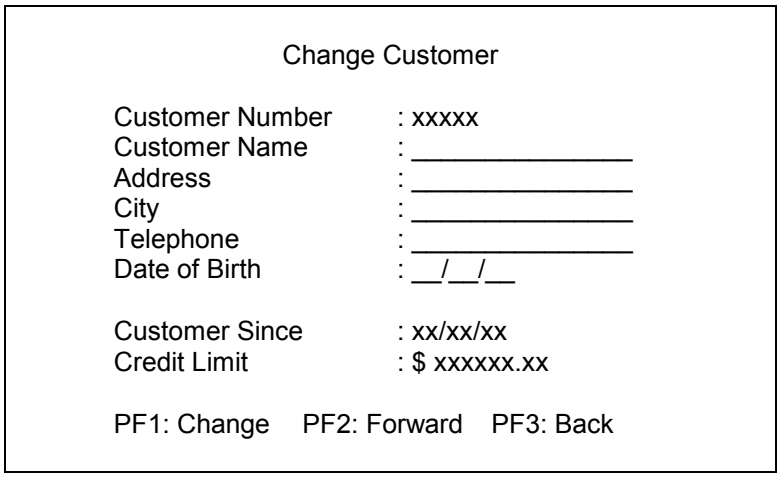
Problem

Using a menu, the user indicates that he wants to change customer data. The application subsequently presents screen 1 on which the user must enter a unique customer number or a customer name (but not both). The change function then has the screen sequence illustrated below.



Screen 1

--`,,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale



Screen 2

After entering one of the two selection criteria, the data of the (first) customer that meets the criterion appears on screen 2. If Customer Number is used as the selection criterion, then only one customer can satisfy the criterion. Function keys are not active in such a case. If Customer Name is used as the selection criterion, however, a number of customers may satisfy the criterion. The user will not receive the kind of overview screen he did in the previous example in order to select a customer, but instead will be able to use the function keys PF2 and PF3 to browse through the customers selected until he has found the one he wants.

Are external outputs or external inquiries counted for this external input (i.e., for the ability to change customer data)? How many functions should be counted in total?

Discussion

If the correct customer data is displayed on the change screen (screen 2) as a result of a customer number having been entered on screen 1, then the display of the data to be changed is part of the external input "Change Customer". The display is not counted as an individual external inquiry.

Counting must be carried out as follows when a user can enter a non-unique customer name on screen 1, browse through the customers on screen 2 until the correct one has been found, and then change the data of the customer. When the user searches for a customer via customer name, this search may result in the selection of a number of customers that can be displayed on the change screen via the function keys. This selection is not fully determined in size beforehand; i.e., the size of the selection varies depending on how many customers meet the selection criteria. The customer desired can then be selected. The display of the selected customers is seen as additional functionality. An additional function is counted because the functionality provided is in fact the same as in the previous illustration in which the selected customers were represented on an overview screen, and because the display and ability to browse through the selected customers entails a different logical processing than when data is merely presented and changed. More concretely, one external output should be counted because the output varies in size.

Sometimes it occurs that an application retrieves only one customer when a user selects data via customer name, in which case the function keys for browsing are not active. Such a situation is considered to be a part of the external output that is counted for selecting. The situation does not result in an additional external output or external inquiry.

The change function is the same in both cases and is therefore counted as one external input.

Solution

Count one external input for the change function.

Count one external output for the ability to browse through the customers that meet the selection criterion.

--`,,```-`-`,,`,,`,`,,`---
© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

References

See section 5.17 and guidelines 8.2.a, 8.2.n, 8.2.x, 8.3.d, 9.2.a, 9.2.c, 9.2.q, 9.2.u, 9.3.a, 9.3.b, and 10.2.j.

11.23 Direct and delayed processing

Problem

An application provides its users the opportunity to update an insurance group for those insured on the basis of region; e.g., regional theft insurance premiums. The application enables its users to assign a zip code series to a different insurance group by means of a screen. In this illustration, three functionally different situations are handled. Each situation shows how function point counting should be carried out.

The update in the first situation takes place immediately after a zip code series has been entered. Then a new series can be entered. When the user is finished with the function and one or more insurance groups have been adjusted, the application prints a report of the transactions for verification purposes.

In the second situation, the user can enter one or more zip code series. The processing of the data is done at night. When the insurance groups have been adjusted, the application prints a report of the transactions for verification purposes. Once the zip code series have been entered, they can no longer be maintained.

In the third situation, the user can enter one or more zip code series. The application processes the data at night, after which it prints a report of the transactions for verification purposes. Now, however, the zip code series entered can still be changed or deleted after they have been entered, but before their nightly processing.

For each of the situations above, determine which functions should be counted. Does the transaction file with the zip code series in situation 2 and/or 3 count as an internal logical file?

11.23.1 Situation 1: Direct processing

Discussion

The main objective of this function is the adjustment of the insurance groups. The data that is saved is functionally permanent data. This means that an external input is present. The creation of the transaction report is inextricably bound to the function, and the report itself fulfills a functional requirement; i.e., it is necessary for verification purposes. The transaction report also crosses the application boundary. For this reason, an external output is counted for the transaction report, even though the function is inextricably bound to the external input.

Solution

Count the following functions:

- One external input for entering and adjusting the insurance groups
- One external output for the transaction report

References

See guidelines 8.2.r and 9.2.p.

11.23.2 Situation 2: Delayed processing

Discussion

The main objective of this function is the adjustment of the insurance groups. The data saved is functionally permanent data. This means that at least one external input is present. FPA considers the entering of the zip

--`,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

code series and the nightly processing of the data as delayed processing. It sees the nightly processing and the entering of the zip code series as a whole.

The zip code series temporarily saved cannot be maintained and are not permanent because the data no longer exists after being processed during the nightly processing. In other words, the data is "consumed". The zip code series therefore form a temporary dataset that cannot be considered an internal logical file.

The creation of the transaction report is inextricably bound to the nightly processing, and the report itself fulfills a functional requirement; i.e., it is necessary for verification purposes. The transaction report also crosses the application boundary. For this reason, the transaction report is counted as an external output, even though the function is inextricably bound to the external input.

Solution

Count the following functions:

- One external input for entering the insurance groups and for the nightly adjustment of the insurance groups
- One external output for the transaction report

References

See guidelines 6.2.g, 8.2.r, and 9.2.p.

11.23.3 Situation 3: Delayed processing and maintenance

Discussion

This main objective of this function is the adjustment of the insurance groups. The data that is saved is functionally permanent data. This means that at least one external input is present. FPA considers the entering of the zip code series and the nightly processing of the data as delayed processing. It sees the nightly processing and the entering of the zip code series as a whole.

The zip code series stored can be maintained and therefore make up an internal logical file. Furthermore, two maintenance functions are counted: one for changing zip code series and one for deleting them.

The creation of the transaction report is inextricably bound to the nightly processing, and the report itself fulfills a functional requirement; i.e., it is necessary for verification purposes. The transaction report also crosses the application boundary. For this reason, the transaction report is counted as an external output, even though the function is inextricably bound to the external input.

Solution

Count the following functions:

- One external input for the initial input of zip code series and the nightly processing together
- One internal logical file for the file containing zip code series
- Two external inputs for the changing and deleting of the zip code series
- One external output for the transaction report

References

See guidelines 6.2.g, 8.2.r, and 9.2.p.

--`,,`,,`-`-`,,`,,`,`,,`---

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.24 Case study of a customer application

Problem

The functional specifications below have been made for a small customer application at an early stage of application development.

The following is maintained for each customer: Name, Address, City, Country Code, Telephone, and Contact Person. The registration numbers of Dutch customers registered at the chamber of commerce (CoC) are also maintained. Users would like to be able to add, change, and delete data. When a user wants to change and delete data, the customer data present must be shown for verification.

Users also want to be able to print the following reports via the menu below:

Reports Menu

1. All Customers
2. Domestic Customers
3. Foreign Customers
4. Contact Persons for Dutch Customers
5. Contact Persons for All Customers

Choice : ___

A sketch of each of these reports is given on the following page. The name of a country is retrieved from a file called Countries that contains the name of a country for each country code. This file is maintained by a different application.

1. Report of "All Customers"

This report contains all customers and is sequenced by company. The country for Dutch customers is not printed.

All Customers

Business Name	Country	Telephone	CoC-nr	Contact Person
AeroDat	Belgium	00-32-2-3456789		Du Spiré
BankBetaal		030-3141592	12345	Westerhof
ImportRossia	Russia	00-7-812-4567890		Ivanets
SehFern AG	Germany	00-49-30-1234567		Strohmann
TevreeConsult		020-7777777	45678	Doeven

2. Report of "Domestic Customers"

This report contains all Dutch customers and is sequenced by company.

Domestic Customers

Business Name	Country	Telephone	CoC-nr	Contact Person
BankBetaal		030-3141592	12345	Westerhof
TevreeConsult		020-7777777	45678	Doeven

3. Report of "Foreign Customers"

This report contains all foreign customers. The user wants the country to appear at the beginning of each line on the list.

Foreign Customers

Country	Business name	Telephone	Coc-nr	Contact Person
Belgium	AeroDat	00-32-2-3456789		Du Spiré
Belgium	LuchtBelga	00-32-81-7654		VandenBerghe
Germany	SehFern AG	00-49-30-1234567		Strohmann
Russia	ImportRossia	00-7-812-4567890		Ivanets

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4. Report of "Contact Persons for Dutch Customers"

The report contains the telephone number and the contact person of all Dutch customers.

Contact persons for Dutch Customers

Business Name	Telephone	Contact Person
BankBetaal	030-3141592	Westerhof
TevreeConsult	020-7777777	Doeven

5. Report of "Contact Persons for All Customers" (by country)

This report contains the telephone number, the chamber of commerce number, and the contact person of all customers. Customers are grouped by country.

Contact Persons for All Customers (by country)

Business Name	Telephone	CoC-nr	Contact Person
Belgium			
AeroDat	00-32-2-3456789		Du Spiré
LuchtBelga	00-32-81-7654		VandenBerghe
Germany			
SehFern AG	00-49-30-1234567		Strohmann
The Netherlands			
BankBetaal	030-3141592	12345	Westerhof
TevreeConsult	020-7777777	45678	Doeven
Russia			
ImportRossia	00-7-812-4567890		Ivanets

Task

Carry out an estimated function point count for this system. In other words, inventory all the functions (logical files and transactions).

Discussion

The entity type Customer can be maintained in the application and is an internal logical file. Country is an FPA table that the application can only read. This is counted as a record type in the FPA tables EIF. Other FPA tables do not exist; therefore, the FPA tables EIF in this case consists of only one record type.

The specifications indicate that customer data can be added, changed, and deleted. This means that three external inputs are counted. The fact that a chamber of commerce number may not be entered for foreign customers does not play a role.

The user has not requested a separate external inquiry. The showing of current customer data for the purpose of verification when a user changes and deletes data is not counted as a separate external inquiry.

Reports 1, 2, and 3 together count as one external output because the following applies in all cases:

- The same object is being reported on (customer)
- The selection criterion is the same (country)
- The processing in order to produce the output products is the same (Except for the selection mechanism, no additional processing is needed.)
- The logical layout of the output products (set of data element types and their structure) is the same; i.e., business name + (country) + telephone + (CoC-nr) + contact person. The parentheses denote optionality. The sequence is not important.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

It is irrelevant that a heading is not printed in all cases, as when data is not present or desired; e.g., a CoC-nr or the name of a country, respectively. The headings, after all, have been defined for the output product.

Although the sequence of the columns is different in report 3, this is no reason to count a separate external output.

In these three cases, a direct selection takes place via the heading Country.

The same result could also be realized with a "fill-in screen" in which the user is provided with country code as a selection criterion.

The fill-in screen would not be counted as a separate external input. Within FPA, the data to be filled in would be considered control information for the external output, and each piece of data would be included in the count as a data element type.

While it is true that report 4 selects the same customers as report 2, the logical layout is different because the set of data element types in 4 is different: business name + telephone + contact person. Report 4 therefore counts as a separate external output.

Report 5 selects the same customers as report 3. The set of data element types is the same in both reports. However, the structure of the output product is different (the data element types are grouped differently) because the country is presented once each time. Therefore the logical layout is different. As a result, report 5 is counted as a separate external output.

According to the guidelines, no transactional functions are counted at all for the FPA tables EIF, even if external inquiries or external outputs would be present.

Solution

A logical file is counted as *low* in an estimated function point count and a transaction as *average*. This results in the following function point count:

Function type	Type	Complexity	Function points	Comments
Customer	ILF	*Low*	7	
FPA tables EIF	EIF	*Low*	5	
Add customer	EI	*Average*	4	
Change customer	EI	*Average*	4	
Delete customer	EI	*Average*	4	
Report 1	EO	*Average*	5	
Report 2	-	-	-	Is the same as report 1 in FPA
Report 3	-	-	-	Is the same as report 1 in FPA
Report 4	EO	*Average*	5	
Report 5	EO	*Average*	5	
Menu	-	-	-	Is not counted
TOTAL			39	

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The size of the system is 39 function points.

References

See sections 4.2.2, 5.20, and 9.1 and guidelines 6.2.l, 8.2.m, 8.2.n, 9.2.w, and 9.3.b.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
(normative)

The most important features and tables for valuing function types

A.1 Internal logical files

Definition

An *internal logical file* is a logical group of permanent data seen from the perspective of the user that meets each of the following criteria:

- It is used by the application to be counted
- It is maintained by the application to be counted

Criteria

- Assume the conceptual data model
- The data must be maintained by the application to be counted
- The data must be functionally permanent
- The data group must be useable, recognizable, comprehensible, and significant to the user

Complexity matrix of internal logical files

The following table is used to determine the complexity level of the internal logical file:

DET RET	1 - 19	20 - 50	51+	
1	L	L	A	L = Low
2 - 5	L	A	H	A = Average
6+	A	H	H	H = High

DET = Data element type

RET = Record type

The number of record types is equal to the number of enclosed entity types.

When the number of record types or the number of data element types of an internal logical file is not yet known, the internal logical file is valued as *low*.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2 External interface files

Definition

An *external interface file* is a logical group of permanent data seen from the perspective of the user that meets each of the following criteria:

- It is used by the application to be counted
- It is not maintained by the application to be counted
- It *is* maintained by a different application
- It is directly available to the application to be counted; i.e., the application to be counted always has the current data from the logical file at its disposal, even though a different application maintains this logical file

Criteria

- Assume the conceptual data model
- The data may not be maintained by the application to be counted
- The data must be functionally permanent
- The data group must be useable, recognizable, comprehensible, and significant to the user

Complexity matrix of external interface files

The following table is used to determine the complexity level of the external interface file:

DET RET	1 - 19	20 - 50	51+	
1	L	L	A	L = Low
2 - 5	L	A	H	A = Average
6+	A	H	H	H = High

DET = Data element type

RET = Record type

The number of record types is equal to the number of enclosed entity types.

When the number of record types or the number of data element types of an external interface file is not yet known, the external interface file is valued as *low*.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.3 External inputs

Definition

An *external input* is a unique function recognized by the user in which data and/or control information is entered into an application from outside that application.

Criteria

- It is an elementary process
- The user specifies it
- It is a unique combination of a set of data element types and logical processing in the application
  - The data crosses the boundary of the application to be counted
  - It usually results in the addition, change, and/or deletion of data in one or more internal logical files

Complexity matrix of external inputs

The following table is used in order to determine the complexity level of an external input:

DET FTR	1 - 4	5 - 15	16+	
0 - 1	L	L	A	L = Low
2	L	A	H	A = Average
3+	A	H	H	H = High

DET = Data element type

FTR = File type referenced

When the number of file type references or the number of data element types of an external input is not yet known, the external input is valued as *average*.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.4 External outputs

Definition

An *external output* is a unique output recognized by the user that crosses the application boundary. It varies in size or further data processing is needed for it.

Criteria

- It is an elementary process
- The user specifies it
- It is a unique combination of a set of data element types and logical processing in the application
- The information distributed crosses the boundary of the application to be counted
- It may include the input of selection criteria or of other control information, but does not necessarily have to
- The output may vary in size
- The output may contain results of arithmetic operations

Complexity matrix of external outputs

The following table is used to determine the complexity level of an external output:

DET FTR	1 - 5	6 - 19	20+	
0 - 1	L	L	A	L = Low
2 - 3	L	A	H	A = Average
4+	A	H	H	H = High

DET = Data element type

FTR = File type referenced

When the number of file types referenced or the number of data element types of an external output is not known, the external output is valued as *average*.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.5 External inquiries

Definition

An *external inquiry* is a unique input/output combination recognized by the user in which the application distributes an output fully determined in size without further data processing, as a result of the input.

Criteria

- It is an elementary process
- The user specifies it
- It is a unique combination of a set of data element types and logical processing in the application
- The information distributed crosses the boundary of the application to be counted
- It should contain the input of selection criteria
- The output may not vary in size
- The output may not contain results of arithmetic operations
- Changes to internal logical files may not occur

Complexity matrix of external inquiries

The following tables are used to determine the complexity level of the external inquiry:

- For the input part, the complexity matrix for external inputs
- For the output part, the complexity matrix for external outputs

Then compare the complexity of the input part and the output part. The complexity of the external inquiry is equal to the complexity of the part with the highest complexity.

When the number of file types referenced or the number of data element types of an external inquiry is not known, the external inquiry is valued as *average*.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.6 Function point table for valuing function types

The following table is used to determine the number of function points of a function type.

	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

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(normative)

Function point analysis glossary

Application: a system for collecting, saving, processing, and presenting data by means of a computer.

Application boundary: the border between the application and its environment; i.e., other applications and users. The application boundary determines the scope for the function point count.

Application function point count: the size of an application expressed in function points; i.e., the functionality already provided to the user or that is still to be provided. With it, the effort required to support the realized application can also be determined.

Automated information system: see application.

Base Functional component: an elementary unit of Functional User Requirements defined by and and used by an FSM Method for measurement purposes.

Complexity of a function: the weight allocated to a function on the basis of which a number of function points is assigned to the function.

Complexity matrix: a table used to allocate a weight to a function type. The matrix allocates this weight on the basis of the number of data element types in combination with the number of record types or file types referenced.

Conceptual data model: a data model that illustrates the data groups as they are seen by the user.

Control information: data that turns on or off one or more processes of an application or that influences the operation of a transaction; e.g., a signal or selection data.

CPM: an abbreviation for Counting Practices International Standard, the IFPUG equivalent of the NESMA International Standard "Definitions and Counting Guidelines for the Application of Function Point Analysis".

Data element type: the most elementary form of data as seen by the user that serves for controlling, recording, or transferring information.

Data function: a logical file; i.e., a logical group of permanent data seen from the perspective of the user. FPA assigns each data function a *type* and distinguishes between the following types: the internal logical file and the external interface file.

Data function type: one of two categories that FPA assigns to a data function: internal logical file and external interface file.

Data information: information that enters or exits the application and that satisfies the user's information need.

Derived data: data that can be derived; e.g., data that has not been recorded in logical files, but that can be derived from the data element types of the logical files.

Detailed function point count: the most accurate count to determine the size of an application or a project in which all the specifications needed for FPA are known in detail. This means that transactions have been specified up to the level of referenced logical files (the so-called file types referenced) 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 of each function recognized can be established.

--`,,`,,`-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Development project: a project in which a completely new application is realized. It entails the specification, construction, testing, and delivery of a new application. During actualization, this project can be split up into a number of sub-projects. If these are carried out more or less in parallel, each being responsible for effectuating a certain sub-system of the total application, then each sub-project should be considered as an individual development project, if the sub-system itself is an application. Re-building an existing application, otherwise known as re-engineering, is considered as development.

Development project function point count: a count that measures a project that provides end-users with the first installation of the software.

Dialog: the conversation between the user and the application needed to execute a transaction.

EI: the abbreviation for external input.

EIF: the abbreviation for external interface file.

Elementary process: a function is an elementary process when two conditions are satisfied:

- The function has an autonomous meaning to the user and fully executes one complete processing of information. In other words, it is self-contained.
- After the function has been executed, the application is in a consistent state.

Enhancement: the activities carried out for an application that change the specifications of the application and that also usually change the number of function points as a result; e.g., change requests.

Enhancement project: a project in which enhancements are made to an existing application. This means that functionality can be added to, changed in, or deleted from an existing application.

Enhancement project function point count: a count that measures a project that realizes modifications to an existing application; e.g., the addition, change, or deletion of functions.

EO: the abbreviation for external output.

EQ: the abbreviation for external inquiry.

Error message: a message that the application gives when incorrect data is entered or when another processing error occurs.

Estimated function point count: a possible function point count in an early phase of an application's life cycle to determine the size of an application or a project in which certain minimum specifications are assumed. Typically the number of functions is recorded per type, and a default value is used for the complexity: *average* for the transactional functions (transactions) and *low* for the data functions (logical files).

External input: a unique function recognized by the user in which data and/or control information from outside the application is entered into the application.

- Data in this context is understood to be data that causes an addition, a change, or a deletion of data in one or more internal logical files.
- Control information is understood to be data (e.g., a signal) that activates or stops one or several processes of the application or that influences the operation of a transaction.

External interface file: a logical group of permanent data seen from the perspective of the user that an application uses but that a different application maintains.

External inquiry: a unique input/output combination recognized by the user in which the application distributes an output fully determined in size without further data processing, as a result of the input.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

External output: a unique output recognized by the user which crosses the application boundary.

File: see logical file.

File type referenced: an internal logical file (ILF) or an external interface file (EIF) maintained or read by a transaction.

Final function point count: a count to determine the number of function points at the end of a project. The count can record the size of a specified or installed application to be maintained, or can determine the size of the project carried out.

FPA: the abbreviation for Function Point Analysis.

FPA table: an entity type that has a secondary function in the application (e.g., code tables, reference tables, entity types with constants, text, or decodings) and whose data can be maintained by the application to be counted or by a different application.

FPA tables EIF: the external interface file that is counted for the set of all FPA tables identified in an application that are only used by the application to be counted, but that are maintained by a different application.

FPA tables ILF: the internal logical file that is counted for the set of all identifiable and maintainable FPA tables in an application.

FTR: the abbreviation for file type referenced.

Function: an elementary unit of requirements/specifications defined and used for measurement purposes.

Function point: a unit which expresses the size of an application or of a project.

Function point analysis: a method used to acquire a measurement of the amount of functionality an application provides a user. The method consists of analyzing the functionality that the application provides or will have to provide. The analysis results in an overview of functions and the complexity of these functions. The measurement is independent of technology and quality requirements.

Function point count: the absolute sum of the number of function points of all the functions to be added to, changed in, or deleted from the project or the application to be counted.

Function point table: a table used to allocate function points to functions, depending on the function type and the complexity established for the function.

Function type: the five types of components, of which an application consists, seen from the perspective of FPA. These components determine the amount of functionality that an application provides to a user. The component types are as follows: external input, external output, external inquiry, internal logical file, and external interface file.

ILF: the abbreviation for internal logical file.

Indicative function point count: an indication denoting the estimated size of an application or project, based exclusively on a conceptual data model or a data model in the third normal-form.

Information system (for FPA): see automated information system.

Initial function point count: a function point count carried out at the beginning of a project.

Interim function point count: a count to determine the size of an interim enhancement during a new development project or an enhancement project; i.e., a count to determine the scope of an addition, a change,

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

or a deletion of functional specifications. Both the change in the application function point count and the project function point count can be the subject of this count.

Internal logical file: a logical group of permanent data seen from the perspective of the user that an application uses and maintains.

List function: an on-line function that displays an overview of entity type occurrences that may or may not have to satisfy a certain selection criterion.

Logical file: a logical group of permanent data seen from the perspective of the user. It is an internal logical file or an external interface file. See also data function.

Logical layout: the set of user required data element types and their logical structure as defined for an output product, apart from aspects of physical implementation. This does *not* pertain to the physical way in which data is presented on a screen, report, or other media.

Menu: a list displayed on a screen showing available functions from which a choice can be made.

Menu structure: the implementation of a dialog by means of a series of interrelated menus and screens.

Number of function points: see function point count.

Output product: the physical form that information can take and that an application distributes. Examples of this include a report, an output file, or a message to a different application.

Project function point count: the size of a development project or an enhancement project expressed in function points. In other words, the total functionality to be added, changed, or deleted. It enables those involved to determine the effort required in order to realize new software or to change the functionality of existing software. In the latter case, a project function point count pertains to the addition, change, or deletion of functions.

Record type: an entity type in a logical file.

Report: an output of data in a layout specified by the user. The output medium used is not relevant for FPA and it can pertain to both an external output and an external inquiry.

System (for FPA): see automated information system.

System table: an entity type that cannot be maintained and, consequently, is not counted within the framework of FPA.

Transaction: see transactional function.

Transaction file: a temporary data file. (It is read one time only and its data is "consumed".)

Transactional function: a transaction. A succession of actions which the user sees as a single work unit. FPA assigns each transactional function a *type* and therefore distinguishes between the following types: external input, external output, and external inquiry.

Transactional function type: one of three categories that FPA assigns to a transactional function: external input, external output, and external inquiry.

Unique function: a function that differs in form and/or logical processing from every other function provided by a certain application.

User: the people, organization(s), end-user(s), functional manager(s), operators, or applications that use or are going to use the application to be measured, as well as the owner or the employees who on behalf of the owner determine the requirements and wishes that have been recorded in the specifications. (These

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

requirements and wishes are established on the basis of the requirements of the end-user(s), for example, but also on the basis of requirements imposed on the application by government or law.)

User perspective: see user view.

User view: the application as seen through the eyes of the user. Crucial here is how the user looks at the exterior of the application in how it relates to supporting the business activities. As a result, it deals with *what* is used and not with the technical tools or the way in which the information processing is realized in a technical sense. It can occur that one function might support several business activities, but that several functions might also support a single business activity.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex C
(informative)

Application of function point analysis including general system characteristics

This Annex explains the application of FPA including general system characteristics.

Section C.1 defines the term “unadjusted function point count” and describes how it is determined. Section C.2 then goes into the general system characteristics and the value adjustment factor. Section C.3 describes the adjusted function point count aqnd its relationship to project budgeting. Section C.4 will first present a generally applicable step-by-step plan.

Sections C.5 and C.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.

C.1 The unadjusted function point count

The unadjusted 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.

C.2 General system characteristics and the value adjustment factor

A general system characteristic is a general characteristic of an application's functional complexity. The degree of influence is expressed on a scale ranging between 0 and 5 and is determined on the basis of user requirements. FPA acknowledges fourteen general system characteristics. The value adjustment factor is computed by using a formula based on the sum of the valuations of each of the fourteen general system characteristics and measures the functional complexity of the user requirements. Its value can vary from 0.65 (low functional complexity) to 1.35 (high functional complexity). Annex D discusses the general system characteristics and the value adjustment factor in more depth.

C.3 The adjusted function point count

The unadjusted function point count (see section C.1) is then multiplied by the value adjustment factor (see section C.2). The product of this calculation is the adjusted function point count, which can therefore deviate up to plus or minus 35% of the unadjusted function point count. The adjusted function point count can then serve as a basis for drawing up a project budget by multiplying it by a productivity norm based on empirical figures. (An example of a productivity norm might be hours per function point.) The actual drawing up of the project budget falls outside the scope of this International Standard, but has been written about in the International Standard *Budgeting on the basis of logical design using function point analysis)*, published by the NESMA.

C.4 Step-by-step plan for carrying out an FPA

Below follows a step-by-step plan for executing 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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Step 2: Determine who the users of the application are. (See section 3.6.)

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 executed, follow the instructions stated in section C.5. If a project function point count must be calculated, follow the instructions found in section C.6.

Step 4: Determine from which other application(s) the application to be counted receives and/or uses data.

Step 5: 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 unadjusted function point count.

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

Step 6: Determine the degree of influence for each of the 14 general system characteristics and record their foundation. (See Annex D.)

Step 7: Determine the value adjustment factor on the basis of the total degree of influence from the 14 general system characteristics.

Step 8: Calculate the number of adjusted function points by multiplying the unadjusted function point count by the value adjustment factor.

Step 9: Together with the user(s), verify the result in relation to those aspects where specific interpretation of the *specifications* available was needed. If necessary, make any corrections as a result of that verification.

Step 10: 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.

C.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.

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.

C.5.1 Determining the application boundary

An *application boundary* is:

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

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

C.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 C.4. 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 C.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 C.5.3.

C.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 unadjusted function points of the application before the change (*UFPB*).

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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 value adjustment factor for the application after the change (*VAFA*).

Step 6: Determine the application function point count of the application after the change (*AFP*) as follows:

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

C.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 C.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 C.5.3.
- The result (the application function point count) of both counting methods is the same.

C.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 C.1 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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

		RELEASE 1	RELEASE 2	RELEASE 3
Unadjusted Function Points	ADD	1000	200	500
	Before CHANGE		80	220
	After		100	200
	DELETE		40	100
Value adjustment factor (VAF)		0.80	0.90	1.00
Project function point count		1000 x 0.80 = 800	Add: +200 Change after: +100 Subtotal: +300 VAF: 0.90 Corrected subtotal: +270 Delete: +40 VAF: 0.80 Corrected Delete: +32 TOTAL: 302	Add: +500 Change after: +200 Subtotal: +700 VAF: 1.00 Corrected subtotal: +700 Delete: +100 VAF: 0.90 Corrected Delete: +90 TOTAL: 790
Application function point count		1000 x 0.80 = 800	Rel. 1: 1000 Add: +200 Change after: +100 Change before: -80 Delete: -40 Subtotal: 1180 VAF: 0.90 TOTAL: 1062	Rel. 2: 1180 Add: +500 Change after: +200 Change before: -220 Delete: -100 Subtotal: 1560 VAF: 1.00 TOTAL: 1560

Figure C.1 — 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.

C.6.1 Determining the scope of the 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 C.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.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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.

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 project boundary:

- 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 project boundary 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 boundary 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.

C.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 C.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.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

C.6.3 The project function point count of enhancement projects[3]

An enhancement project considers enhancements to an existing application. This means that functionality can be added to, changed in, and deleted from this application.

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 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 and establish how many function points they represent (*ADD*).

Step 4: Determine the value adjustment factor for the application after the change (*VAFA*).

Step 5: Determine the value adjustment factor for the application before the change (*VAFB*).

Step 6: Calculate the project function point count for the enhancement project as follows (*EFP*):

EFP = DEL * VAFB + (CHGA + ADD) * VAFA

Step 7: 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 6. Do this for any adaptations that are necessary in other applications too.

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.

C.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 C.6.2.)

[3] 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.

--`,,,```-`-`,,`,,`,`,,`---

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex D
(informative)

General system characteristics

An application can be characterized by fourteen general system characteristics in FPA. These characteristics are a result of functional requirements and determine the processing complexity of the application as a whole. A weight can be allocated to each general system characteristic in order to express a degree of influence. The processing complexity is expressed in a value adjustment factor for the general system characteristics and by totaling the degrees of influence linked to each characteristic. The unadjusted function point count is converted into an adjusted function point count with this value adjustment factor.

Each general system characteristic is discussed individually in the sections below. First a description is given, and then a number of guidelines to determine the degree of influence. If none of the guidelines applies exactly to the application, the counter must determine which degree of influence is most fitting.

The value adjustment factor is determined in two steps:

1. By evaluating each of 14 general system characteristics degree of influence that, together, determine the total degree of influence
2. By calculating the value adjustment factor

The degree of influence of a general system characteristic is expressed on a scale of 0 to 5. See the descriptions for determining the degree of influence in the sections below.

General system characteristics

1. Data communications
2. Distributed data processing
3. Performance
4. Heavily used configuration
5. Transaction rate
6. On-line data entry
7. End-user efficiency
8. On-line update
9. Complex processing
10. Re-usability
11. Installation ease
12. Operational ease
13. Multiple sites
14. Facilitate change

The second step, calculating the value adjustment factor, is done as follows:

Value adjustment factor = 0.65 + 0.01 * (*total degree of influence*)

The *total degree of influence* is the sum of the degrees of influence allocated to the general system characteristics.

The value adjustment factor can therefore increase or decrease the number of unadjusted function points by as much as 35%.

D.1 Data communications

The data and control information used in the application are sent or received via data communication facilities.

A group of terminals connected locally to a computer system is not considered a data communication facility. Data communication bridges a geographical distance and, generally speaking, requires special facilities. If such facilities are only a more-or-less incidental part of the technical infrastructure selected, it should not be included in the valuation.

--`,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2005 All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Degrees of influence:

0: The application is solely batch processing or stand-alone PC processing.

1: The application is batch, but has remote data entry *or* remote printing only.

2: The application is batch, but has remote data entry *and* remote printing.

3: On-line functions form mainly a front-end to a batch process or query system.

4: The application is more than a front end (e.g., a combination of on-line and possibly batch processing), but supports only one type of communication protocol for teleprocessing.

5: The application is more than a front-end and supports several kinds of communication protocols for teleprocessing.

D.2 Distributed data processing

The application includes distributed data or distributed processing functions.

Distributed data means that files are dispersed over the components of the application.

Distributed processing functions means that information processing is divided among the components of the application.

A component is a technological unit, a computer configuration, a separate CPU, a database processor, an end-user computer system, and so on.

Degrees of influence:

0: The application does not aid the transfer of data or processing functions between components of the application.

1: The application prepares data for end-user processing on another component of the system; e.g., PC spreadsheets and PC DBMSs.

2: Data is prepared for transfer, sent off, and processed on another component of the system (not meant for use by the end-user). The start up and processing of the transfer is done International Standardly.

3: Distributed processing and data transfer occur on-line and in one direction only. The start up and processing of the transfer is done International Standardly.

4: Distributed processing and data transfer occur on-line and can occur in both directions. Starting up and processing the transfer is done International Standardly.

5: Processing functions are performed dynamically on the most appropriate component of the system.

D.3 Performance

The performance objectives of response times or throughput times that a user has determined or approved influence the development, implementation, and maintenance of the application.

--`,,,```-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
© ISO/IEC 2005 – All rights reserved
Not for Resale

Degrees of influence:

0: No special performance requirements were stated by the user; e.g., "quick and dirty" solutions, such as applications with a planned, short life cycle.

1: Response time requirements have been stated by the user, but no additional effort is required.

2: The on-line response time is critical during peak hours. No special design for CPU utilization is necessary. The following business day is the processing deadline.

3: The on-line response times are critical during all business hours. No special design for CPU utilization is required. Other applications with which the application interfaces impose requirements on the processing deadline.

4: The response times the user requires are of such a nature that performance analysis activities have been included in the design phase.

5: Performance analysis activities and tools are used in the design, development, or implementation phases of the application in order to satisfy the performance objectives the user has specified.

D.4 Heavily used configuration

The application must operate on a computer system that is already heavily used (heavy use during operation) and therefore requires specific design considerations.

Degrees of influence:

0: There are no explicit or implicit operational restrictions.

1: Operational restrictions do exist but are less restrictive than usual. No additional effort is needed to meet the restrictions.

2: There are some security or timing considerations included.

3: Specific requirements for a specific part of the application have been imposed on the production system.

4: Operational restrictions impose special constraints on the structure of the application.

5: Operational restrictions impose special constraints on the structure of the application, and specific constraints are imposed on other distributed components of the application.

D.5 Transaction rate

Transaction rate refers to the number of transactions processed per unit of time. Essential here is the degree to which the transaction rate is relevant to the architecture of the application. Keep peak loads particularly in mind. A high transaction rate influences the development, implementation, and maintenance of the system. A low transaction rate has no influence on the development, implementation, or maintenance of the system.

Degrees of influence:

0: No attention is paid to performance analysis.

1: Attention is paid to performance analysis, but the transaction rate is very low.

2: The transaction rate is normal, but the transactions are fairly simple, so that performance analysis does not play a large role.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3: The transaction rate and the complexity are of such a nature that performance analysis considerations are standard.

4: Performance analysis tasks make up part of the design phase of the project in order to be able to satisfy the high transaction rate specified by the user.

5: Performance analysis tasks are executed and performance analysis aids are used in the design, development, or installation phase of the application.

D.6 On-line data entry

The application provides on-line data entry (input of data) or on-line functions for the input of control information. This involves *kinds* of transactions, not the rate at which the transactions are carried out.

Degrees of influence:

0: There are no interactive data-entry transactions.

1: 1-7% of the transactions are interactive data entry.

2: 8-15% of the transactions are interactive data entry.

3: 16-23% of the transactions are interactive data entry.

4: 24-30% of the transactions are interactive data entry.

5: More than 30% of the transactions are interactive data entry.

D.7 End-user efficiency

The transactions in the application are geared towards attaining high efficiency from the end-user. Issues here pertain to the structure of the application that will make an end-user more efficient in the way he works with the application in his organization.

The following can be considered:

- Menus
- On-line help functions and help documentation
- Automated cursor movement
- Scroll functions
- Remote printing (via on-line transactions)
- Pre-assigned function keys
- Submission of batch jobs from on-line transactions
- Cursor selection of screen data
- Heavy use of reverse video, highlighting, colors, and other indicators
- Making a hard copy of on-line transactions

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- Interface for the use of the mouse
- Pop-up screens
- The use of as few screens as possible to execute a business function
- Easy navigation between screens (e.g., via function keys)
- The support of several languages:
    - Two languages: count as four items
    - Several languages: count as six items

Degrees of influence:

0: None of the above appear.

1: One, two, or three of the above appear.

2: Four or five of the above appear.

3: Six or more of the above appear, but there are no specific user requirements regarding efficiency.

4: Six or more of the above appear, and user requirements regarding efficiency are of such a nature that special tasks must be developed for them.

5: Six or more of the above appear, and user requirements regarding efficiency are of such a nature that special resources and processes are needed to demonstrate that the objectives have been achieved.

D.8 On-line update

On-line update of the application's internal logical files occurs. Involved here are the *kinds* of transactions, not the rate at which the transactions are carried out.

Degrees of influence:

0: Not applicable.

1: Files with control information are updated on-line. The rate at which this occurs is low and recovery does not occur.

2: Internal logical files are updated on-line. The rate of this is low and recovery is simple.

3: Important internal logical files are updated on-line. Internal control requirements also play a role.

4: Same as 3. In addition, protection against data loss is an essential feature.

5: Same as 4. In addition, cost aspects play a role in recovery considerations, in connection with high volumes.

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

D.9 Complex processing

The internal processing of the application is complex. This may be the case if there are:

- extensive control or security facilities
- extensive logical processing
- extensive mathematical processing
- many exception routines in which transactions must be interrupted and restarted
- complex processes to handle multiple input and output possibilities; e.g., multi-media and device independence

Degrees of influence:

0: None of the features above is present.

1: One of the features above is present.

2: Two of the features above are present.

3: Three of the features above are present.

4: Four of the features above are present.

5: All five of the features above are present.

D.10 Re-usability

The application is structured and maintained in such a way that parts of it can be used by the same application or by another application. Re-usability revolves around software or parts of software that can be used as a functional unit outside the application.

Degrees of influence:

0: The production of code for re-use is not taken into account.

1: The production of code for re-use *in the application concerned* is taken into account only.

2: Code suited for re-use *in other applications* is produced in less than 10% of the modules to be developed.

3: Code suited for re-use *in other applications* is produced in 10% or more of the modules developed.

4: The entire application is structured or documented in such a way that produced code can be re-used easily. In the event of re-use, you must adapt the code to the level of source code.

5: The entire application is structured or documented in such a way that the re-use of produced code is simple. In the event of re-use, you can adapt the code via user parameters.

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

D.11 Installation ease

Ease of installation is characterized by:

- the convenience with which the application can be installed and
- the conversion required for this

A conversion and installation plan or conversion tools are made and tested during the test phase of the application concerned.

Degrees of influence:

0: There are no plans pertaining to installation and conversion.

1: The user has not stated any plans, but a special setup is required for installation.

2: The user has stated requirements for conversion and installation, and guidelines for the installation have been made and tested. The conversion does not have any significant impact on the project.

3: The user has stated requirements for conversion and installation, and guidelines for the installation have been made and tested. The conversion has a large impact on the project.

4: As in (2), but automated tools for the installation and conversion are also required and tested.

5: As in (3), but automated tools for the installation and conversion are required and tested.

D.12 Operational ease

The application's operational ease is characterized by requirements for decreasing the International Standard effort required to operate the application (operation and Computer Center). This can include computerization (preventing International Standard activities) or the setup of procedures for:

- starting up the application
- backing up data
- recovering the application
- tape handling
- paper handling

Degrees of influence:

0: The user has not formulated any other operational requirements than the standard backup procedures.

1-4: Choose from the following items that apply to the application concerned. Count one point for each item unless directed otherwise:

- Processes have been made for starting up the application, backing up data, and recovering the application, but the operator is required to intervene.

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- Processes have been made for starting up the application, backing up data, and recovering the application in which the operator is *not* required to intervene. (Count as two items.)
- The application restricts to a minimum the number of times that a tape is to be set up.
- The application restricts to a minimum the number of times that other paper is to be put in the printer.

5: The requirements are as such that the application must be set up for unattended operation. In other words, no operator intervention should be required other than to start up or shut down the application. The application has an automatic recovery facility.

D.13 Multiple sites

The application is specifically designed, developed, and supported to facilitate implementation at multiple organizations or multiple sites.

Degrees of influence:

0: The application is meant for one user at one site.

1: During design, the use of the application at multiple sites is taken into account, but only in identical hardware and software environments.

2: During design, the use of the application at multiple sites is taken into account, but then in similar hardware and software environments.

3: During design, the use of the application at multiple sites is taken into account, with different hardware or software environments.

4: Documentation and a support plan are made and tested to support the application at multiple sites, and the application is as described in (1) or (2).

5: Documentation and a support plan are made and tested to support the application at multiple sites, and the application is as described in (3).

D.14 Facilitate change

There are specific requirements relating to the flexibility of the application. Flexibility is the extent to which the user can influence the operation of the application remotely without technological intervention.

Degrees of influence:

0: No special requirements have been formulated.

1-5: Determine which of the following items apply to the application. Count one point for each item, unless indicated otherwise.

- A flexible query facility is provided that can handle simple query requests; e.g., "and/or logic" applied to one logical file only.
- A flexible query facility is provided that can handle query requests of average complexity; e.g., "and/or logic" applied to several logical files. (Count as 2 items.)

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- A flexible query facility is provided that can handle complex query requests; e.g., "and/or logic" combinations applied to several logical files. (Count as 3 items.)

- Control information is kept in database tables and is maintained on-line by the user. Changes, however, take effect on the next business day only.

- Control information is kept in database tables and is maintained on-line by the user. Changes take effect immediately. (Count as two items.)

--`,,```-`-`,,`,,`,`,,`---

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Alphabetical Index

Adjusted function point count, 106, 107
Adjusted number of function points
  see adjusted function point count, 106
Analysis
  moment of the count, 22
  objective of the count, 22
  required documentation, 22
  type of the count, 22
Application
  definition, 12, 108
  determining the application function point count, 11, 107
Application boundary, 5
  definition, 12, 107
  guidelines, 12, 108
Application function point count, 4, 11, 107
Authorization, 26, 55
Automated information system, 12, 108
Averages, 50
Bachmann diagram, 10
Backup, 119
Batch program, 116
Browsing, 28, 49, 53
  practical situation, 81, 82
Change screen, 44
Check-point data file, 36
Cleaning functions, 28
Colors, 116
Column descriptions, 50
Combination effects of functions, 49
  practical situation, 77
Combined external outputs, 72
Completeness check on the function point count, 29
Complex processing, 118
Complexity matrix, 6
  for an external input, 45
  for an external inquiry, 54
  for an external interface file, 40
  for an external output, 51
  for an internal logical file, 37
Complexity of a function
  definition, 6
  levels of, 6
Conceptual data model, 36, 38
Constants, 29, 37, 39, 50
Construction
  moment of the count, 23
  objective of the count, 24
  required documentation, 24
  type of the count, 23
Continuation screens, 44
Control information
  definition, 41, 101
  in an external input, 43
  in an external output, 47, 50
  practical situation, 67
Conversion, 15, 111, 119
  practical situation, 69
Conversion software, 15, 111
Conversion table normalized entity types to LFs, 33
Counting from a logical perspective, 25
Counting from screens, 19
Counting of
  applications, 11, 107
  development projects, 15, 111
  enhanced applications, 13, 108
  enhancement projects, 16, 112
  external inputs, 42
  external inquiries, 52
  external interface files, 38
  external outputs, 47
  GUI environments, 20
  internal logical files, 36, 61
  new applications, 12, 108
  packages, 17
  traditional design, 17
  transaction files, 66
Cursor selection, 116
Data, 41, 101
Data communications, 113
Data element types
  address data, 50
  control information, 44, 50
  error messages, 50
  fixed data, 50
  function keys, 44, 50
  input screen several functions, 44
  of an external input, 44
  of an external inquiry, 53
  of an external interface file, 40
  of an external output, 50
  of an internal logical file, 37
  process data, 50
  quantity on a report, 72
  selection or control data, 50
  standard data, 50
  with normalization, 37, 40
Data Flow Diagram, 17
Data function type
  definition, 6
Data model in the third normal-form, 30
Datalogical model, 22
Decodings, 29, 37, 39
Delayed processing, 89
Denormalization, 31
  practical situation, 59, 61
Detailed function point count
  applicability, 10
  definition, 10
  during construction, 23
  during functional design, 23
  during operation and maintenance, 24
  specifications required, 10
Development project
  conversion software, 15, 111
  definition, 14, 110
  steps to determine the project function point count, 15, 111
Direct and delayed processing, 89
Direct processing, 89
Distributed data processing, 114
Double counting, 25
End-user efficiency, 116
Enhancement, 13, 15, 108, 111
Enhancement project
  definition, 14, 111
Enhancement project
  steps to determine the project function point count, 16, 112
Entity dependence, 31
Entity independence, 31
Entity Relationship Diagram, 10, 17
Error messages, 27, 48, 57
Estimated function point count
  applicability, 9
  definition, 9
  during analysis, 22
  specifications required, 10
Exception routines, 118
External input, 6
  automatic displaying of data, 43
  combined external inputs, 65
  complexity matrix, 45
  data element types, 44
  data to control an external output, 44
  definition, 41
  delay in time, 43
  determining the complexity of, 44
  duplicate of another external input, 42
  elementary process, 41
  entering data in two steps, 42
  for technological reasons, 43

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

guidelines, 40
logical files, 44
number of data element types unknown, 45
number of referenced logical files unknown, 45
several functions on one screen, 42
several input screens, 43
the lack of, 42
via different media, 43
External inquiry, 6
compared to an external input, 52
compared to an external output, 53
complexity matrix, 54
data element types, 53
definition, 51
determining the complexity of, 53
elementary process, 52
input part, 53
logical files, 53
number of data element types unknown, 54
number of referenced logical files unknown, 54
output part, 53
query facility, 53
External Inquiry
guidelines, 51
External interface file, 6
complexity matrix, 40
data element types, 40
definition, 38
deriving from a normalized data model, 30
determining the complexity of, 40
FPA tables EIF, 39
guidelines, 38
number of data element types unknown, 40
record types, 40
External output, 6
as a result of the technology used, 48
combined external outputs, 72
complexity matrix, 51
data element types, 50
data for controlling of, 47
data intended for other applications, 48
definition, 45
determining the complexity of, 50
elementary process, 47
error messages, 48
guidelines, 45
logical files, 50
number of data element types unknown, 51
number of referenced logical files unknown, 51
on the basis of selection criteria, 49
output file with transactions, 48
output part of an external inquiry, 47
output product, 45
overview of possible error messages, 48
referenced data element types, 50
several external outputs, 48
sorted in several ways, 48
transaction report or processing report, 48
with summary information, 70
Facilitate change, 120
File
historical, 36
interim, 36
process data, 37
technical files, 36
temporary, 36, 45
Final function point count, 11
FPA
background and objective, 3
budgeting projects, 3
during analysis, 22
during counstruction, 23
during functional design, 23
during implementation, 24
during operation and maintenance, 24
during requirements phase, 21
measuring productivity, 3
rationale, 4
reliability, 10
step-by-step plan for carrying out, 8, 106
FPA tables, 29, 58
FPA tables EIF, 29, 39
FPA tables ILF, 29, 37
practical situation, 58, 61
Function (types)
definition, 6
table for valuing, 7
valuing, 7
Function keys, 44, 50, 116
Function point, 3
Function point count, 4, 7, 8
adjusted, 106, 107
detailed, 10
estimated, 9
final, 11
indicative, 8
initial, 11
interim, 11
of a project, 4, 13, 109
of an application, 4, 11, 107
unadjusted, 106, 107
Function point table, 7
Functional design
moment of the count, 23
objective of the count, 23
required documentation, 23
type of the count, 23
General system characteristic, 106, 113
complex processing, 118
data communications, 113
distributed data processing, 114
end-user efficiency, 116
facilitate change, 120
heavily used configuration, 115
installation ease, 119
multiple sites, 120
on-line data entry, 116
on-line update, 117
operational ease, 119
performance, 114
re-usability, 118
transaction rate, 115
Graphical User Interface, 20
Graphs, 27
GUI environments, 20
Hard copy, 116
Header record, 43, 49
Heavily used configuration, 115
Help documentation, 116
Help facilities, 27, 116
practical situation, 56
Highlighting, 116
Historical files, 36
IFPUG, 3
Implementation
moment of the count, 24
training, 24
type of the count, 24
writing the user International Standard, 24
Indexes, 36, 39
Indicative function point count
applicability, 9
definition, 8
during requirments phase, 21
specifications required, 9
Infological model, 22
Information flows, 10
Initial function point count, 11
Input records, 26
Installation, 119
Installation ease, 119
Interim files, 36
Interim function point count, 11
Internal logical file, 6
complexity matrix, 37
data element types, 37
definition, 35
deriving from a normalized data model, 30
determining the complexity of, 37
FPA tables ILF, 37
guidelines, 35

© ISO/IEC 2005 All rights reserved

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

number of data element types unknown, 38
practical situation, 59, 61
record types, 37
Intersection entity
see key-key entity, 31
Key-key entity, 31
Languages, 49, 117
Layout
see logical layout, 46
Less or more work
determining, 22, 23, 24
List functions, 28
practical situation, 81
Literals, 50
Log report, 28
Logical file
see data function type, 6
see external interface file, 6
see internal logical file, 6
Logical layout, 46
practical situation, 73
Logical perspective, 25
Log-on functions, 26
practical situation, 55
Mathematical processing, 118
Menu structures, 28, 116
practical situation, 57
Message headers, 50
Messages, 27
display on seperate screen, 44
Mouse, 117
Multiple sites, 120
Normalization
deriving the logical files, 30
Number of adjusted function points
see adjusted function point count, 106
Number of function points
see Function point count, 7
Number of unadjusted function points
see unadjusted function point count, 106
On-line data entry, 116
On-line update, 117
Operating systems, 26
Operation and maintenance
moment of the count, 24
objective of the count, 24
required documentation, 24
type of the count, 24
Operational ease, 119
Operational restrictions, 115
Operator, 46
Output product, 45
Output records, 26
Packages, 17
determining logical files, 18
determining the functionality required, 18
determining transactions, 18
Page number, 50
Paper handling, 119
Performance, 114
Performance analysis, 115
Pop-up screens, 117
Print files, 36
Processing complexity, 113
Processing report, 48
Productivity
of the development environment, 24
Productivity attributes, 1
Productivity standards, 1
Project boundary
guidelines, 15, 111
Project budgeting, 1
Project function point count, 4, 13, 109
Project scope
definition, 14
Projects
development projects, 14, 110
enhancement projects, 14, 16, 110, 112
to replace an application, 16, 112
Prototyping, 20
Query facility, 27, 53, 120
practical situation, 56
Querying
with different search keys, 79
Record types
of external interface files, 40
of internal logical files, 37
with normalization, 33
Recovery, 119
Re-engineering, 16, 112
Remote printing, 116
Replacing an application
application function point count, 13, 109
project function point count, 16, 112
Report generators, 27
practical situation, 56
Reports, 26
on different media, 67
Requirements phase
moment of the count, 21
objective of the count, 21
required documentation, 21
type of the count, 21
Response times, 114
Restart-recovery mechanism, 36
Results of calculations, 50
Re-usable code, 118
production of, 26
Re-use of existing code, 26
Reverse video, 116
Running register, 37
Scope creep
in analysis, 22
in requirements phase, 21
Scope of the count, 5
definition, 5, 110
Screens, 26
counting from, 19
with a list function, 81
Scroll functions, 116
practical situation, 82
Security, 26, 115, 118
Selection screens, 28, 49
practical situation, 82, 85
Shared use of data, 34
Sort files, 36
Sorting, 48
Spool files, 36, 48
Starting up the application, 119
Subtotals, 50
Systelogical model, 22
System boundary
see application boundary, 5
System date, 50
System scope
see Scope of the count, 5
System table, 29
Tables
see FPA tables, 29
Tape handling, 119
Technological model, 22
Temporary files, 36, 45
Third normal-form, 30, 36, 38
Throughput times, 114
Totals, 50
Trailer record, 43, 49
Transaction file, 45
practical situation, 66
Transaction rate, 115
Transaction report, 48
Transactional function type
definition, 6
Transactions, 38, 42
from another application, 43
on-line data entry, 116
Unadjusted function point count, 106, 107
Unadjusted function point table, 100
Unadjusted nuber of function points
see unadjusted function point count, 106
User interface, 20
Users
definition, 5
Utilities, 26
Value adjustment factor, 106, 107
Wild card, 46
Work files, 36

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ICS 35.080

Price based on 124 pages

© ISO/IEC 2005 – All rights reserved
Copyright International Organization for Standardization
Reproduced by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale