Function Point Counting Practices Manual

Release 4.2

International Function Point Users Group (IFPUG)

Function Point Counting Practices Manual

Release 4.2

Chairperson, Counting Practices Committee
Valerie Marthaler
The David Consulting Group
Clarkston, Michigan

ISBN 0-963-1742-9-0

Release 4.2, January 2004

This release replaces Release 4.1.1, which is now obsolete.
Changes are made periodically to the information within.

Documentation Team

Subject Experts and Writers

Bonnie S. Brown, EDS

Martin D'Souza, Holistic Software Metrics

E. Jay Fischer, JRF Consulting, Inc.

Dave Garmus, The David Consulting Group

Valerie Marthaler, The David Consulting Group

Koni Thompson Houston, Quality Dimensions, Inc.

Adri Timp, Interpay Nederland

Eddy van Vliet, Chameleon Solutions

For information about additional copies of this manual, contact

IFPUG

191 Clarksville Road

Princeton Junction, NJ 08550

U.S.A.

(609) 799-4900

E-mail: **ifpug@ifpug.org**

Web: http://www.ifpug.org

Table of Contents

	Preface	**iii**
	Introduction to the Counting Practices Manual	**vii**
Part 1	**Process and Rules**	
Chapter 1	**Introduction**	**1-1**
Chapter 2	**Overview of Function Point Analysis**	**2-1**
Chapter 3	**User View**	**3-1**
Chapter 4	**Determine Type of Count**	**4-1**
Chapter 5	**Identify Counting Scope and Application Boundary**	**5-1**
Chapter 6	**Count Data Functions**	**6-1**
Chapter 7	**Count Transactional Functions**	**7-1**
Chapter 8	**Determine Value Adjustment Factor**	**8-1**
Chapter 9	**Calculate Adjusted Function Point Count**	**9-1**
	Index	**i-1**
Part 2	**Counting Practices**	
	Introduction	**1-1**
Chapter 1	**Code Data**	**1-3**
Chapter 2	**Logical Files**	**2-1**
Chapter 3	**Shared Data**	**3-1**
Chapter 4	**Enhancement Projects and Maintenance Activities.**	**4-1**
	Index	**i-1**

Part 3 Examples

	Introduction……..	1-1
Chapter 1	Data Function Counting Examples…..…….	1-3
Chapter 2	Transactional Function Counting Examples ..……..	2-1
Index		i-1

Part 4 Appendices and Glossary

Appendix A	Calculation Tables ……	A-1
Appendix B	The Change from CPM 4.1.1 to 4.2 …..……	B-1
Appendix C	Reader Request Form……...……..…….....	C-1
Glossary		G-1

Preface

Introduction

The use of function points, as a measure of the functional size of software, has grown in the past decade from a few interested organizations to an impressive list of organizations worldwide.

IBM CIS & A Guidelines 313

In the late 1970s, Allan Albrecht of IBM defined the concepts that enabled measuring the output of software development projects. These definitions were extended in *IBM CIS & A Guideline 313, AD/M Productivity Measurement and Estimate Validation*, dated November 1, 1984.

Release 2.0

With the growth in the use of function points, there was wider and wider application of the measure. This broadening of the application tested the original description of the measure and made it necessary to create guidelines to interpret the original rules in new environments. This was reflected in Release 2.0 (April 1988) of the *International Function Point Users Group (IFPUG) Function Point Counting Practices Manual.*

Release 3.0

Release 3.0 (April 1990) of the *IFPUG Function Point Counting Practices Manual* was a major milestone in the evolution of functional size measurement. For the first time, the IFPUG Counting Practices Committee made an effort to change the document from a collection of many interpretations of the rules to a truly coherent document that represented a consensus view of the rules of function point counting. In this sense, it was the first step to truly establishing standards for function point measurement which could be applied across organizations.

Release 4.0 Release 4.0 (January 1994) was the next milestone in the evolution of functional size measurement. This release reflected the use of function points early in project development to estimate project size using information engineering disciplines. The rapidly increasing number of graphical user interface (GUI) windows applications mandated that we include GUI counting in the release. Because more counting was occurring across a wider variety of situations, the release placed an emphasis on interpreting and practicing using the counting rules. Examples were included throughout the documentation and case studies supplemented the material. Finally, release 4.0 continued to clarify and increase the consistency of function point counting.

Release 4.1 Release 4.1 (January 1999) provided clarifications to existing rules, new or amended rules which address previously undocumented situations and new hints and examples to aid understanding. The IFPUG Counting Practices Committee has reviewed and processed requests from members, following the Manual Revision Process contained in Chapter 1 of this manual.

The revisions included in 4.1 clarify:

- the identification of a user, an elementary process, and control information
- the differentiation between External Outputs (EOs) and External Inquiries (EQs)
- the identification of Data Element Types (DETs) and Record Element Types (RETs) for data functions
- the identification of Data Element Types (DETs) for transactional functions

Release 4.1 continues the process of clarifying and improving the consistency of function point counting.

Finally, with the exception of the 14 General Systems Characteristics, it was designed to be compliant with existing ISO standards if and when any compliance guide becomes a standard.

Release 4.1.1 Release 4.1.1 (April 2000) corrected a small number of typos and errors.

Release 4.2

Release 4.2 does not modify any previously promulgated rules, but it provides clarification and enhanced interpretations for existing rules that will further increase inter-counter consistency.

The IFPUG function point analysis (FPA) process and rules are concise and easy to use. To reflect that, and to make the Counting Practices Manual (CPM) even more attractive as a reference manual, the Counting Practices Committee (CPC) decided to restructure CPM 4.2 into four parts:

1. Process and Rules
2. Counting Practices
3. Examples
4. Appendices

To assist worldwide practitioners of FPA in a timely manner, the CPC initially published the results of its research as separate publications, addendums to the CPM:

- Counting Logical Files (September 2001)
- FPA in an Enhancement Environment (April 2002)
- Counting Code Data (September 2003)
- Counting Shared Data (September 2003)

These documents have now been incorporated as chapters in Part 2 of the CPM.

The CPC believes that CPM 4.2, with its added guidelines and examples will ensure more consistent results between Certified Function Point Specialists.

Future Releases

This document is meant to be a living one. We must recognize how to count new environments as they are introduced. We need to be able to do this in the context of maintaining the validity of the counts we have already made. This will not be an easy task, yet it is an essential one if we are to be able to measure the progress we are making in delivering value to the users and to the organizations they represent.

The Counting Practices Committee wishes to thank all those who have helped us in our research and in the production of this manual.

Valerie Marthaler
Chairperson, Counting Practices Committee

This page intentionally left blank.

Introduction to the Counting Practices Manual

Introduction

This introduction defines the objectives of the manual and the revision process. It also describes publications that are related to this manual.

Contents

This chapter includes the following sections:

Topic	See Page
Objectives of the Counting Practices Manual	viii
Documents Used for Release 4.2	viii
Intended Audience	ix
Organization of the Counting Practices Manual	x
Manual Revision Process	xi
Frequency of Changes	xi
Change Process	xi
Related IFPUG Documentation	xiv
Training Requirements	xvi

Objectives of the Counting Practices Manual

The primary objectives of the *IFPUG Counting Practices Manual, Release 4.2,* are to

- Provide a clear and detailed description of function point counting
- Ensure that counts are consistent with the counting practices of IFPUG affiliate members
- Provide guidance to allow function point counting from the deliverables of popular methodologies and techniques
- Provide a common understanding to allow tool vendors to provide automated support for function point counting

Documents Used for Release 4.2

The following documentation was used to develop this release:

- The IFPUG FPA method is based on *IBM CIS & A Guideline 313, AD/M Productivity Measurement and Estimate Validation*, dated November 1, 1984. The function point counting methodology described in 313 is generally referred to as Albrecht 1984.
- The current version of the manual, CPM 4.2, is based primarily on the *IFPUG Function Point Counting Practices Manual, Release 4.1.1.*
- Except for the general system characteristics CPM 4.2 is designed to be compliant with ISO/IEC 14143-1:1998 *Information technology – Software measurement – Functional size measurement – Definition of concepts*.
- "*Framework for Functional Sizing*"; this IFPUG paper explains that product size contains three dimensions: functional size, technical size and quality size. The IFPUG FPA-method provides a measure for the functional size.
- Issues not sufficiently covered in the sources listed above were decided by the IFPUG Counting Practices Committee based on variants of existing counting practices and validated through impact studies.

With its release, this manual should be considered the IFPUG standard for function point counting. It is imperative that each IFPUG member take an active role to ensure counting consistency. IFPUG member adherence to this standard will contribute greatly to counting consistency.

Intended Audience

The standard in this manual should be applied by anyone using function point analysis for functional size measurement. The manual was designed for use by persons new to function point counting as well as those with intermediate and advanced experience.

Organization of the Counting Practices Manual

There are four major parts in the Counting Practices Manual (CPM):

- Part 1: Process and Rules
- Part 2: Counting Practices
- Part 3: Examples
- Part 4: Appendices and Glossary

Part 1: Process and Rules includes the IFPUG Function Point Analysis (FPA) method, the process, rules, and some additional hints and guidelines.

To speak a language as a native, learning the grammar and the words alone are not sufficient. They just provide a framework. You need language experience to understand how the language is spoken in practice, how the grammar rules should be applied, what idiomatic expressions are common, and so on. The same is true for FPA. The knowledge of process and rules, as reflected in Part 1, is a necessity, but the knowledge alone is not a sufficient condition to apply FPA correctly. That's why the CPM contains **Part 2**, **Counting Practices** and **Part 3 –Examples**.

Detailed examples are provided extensively in Parts 2 and 3 to explain counting practices concepts and rules. Each example should be considered on its own merits. Since each example is intended to illustrate a specific scenario, variations may exist between examples. Although the examples throughout the manual deal with similar subject matter, they are not intended to represent a single set of user requirements.

Part 4: Appendices and Glossary contains valuable additional information, such as ready to use calculation templates, the transition from CPM 4.1 (and CPM 4.1.1) to CPM 4.2, and the glossary.

In principle, each part stands on its own.

Manual Revision Process

This section explains the frequency of changes to the Counting Practices Manual and defines the change process.

Frequency of Changes

During January of each year, a new version of the Counting Practices Manual *may* become effective. It will include any new or changed definitions, rules, or counting practices that have been finalized by the Counting Practices Committee (CPC) since the previous version.

Change Process

The following activities outline the process for adding or changing information in the Counting Practices Manual. Explanations of each activity follow the table.

Step	Action
1	The issue is submitted to the CPC.
2	The issue is assigned for research.
3	The CPC reviews and discusses the issue.
4	The CPC presents a proposed solution to the IFPUG membership.
5	An impact study is initiated if the proposed change would have any impact on existing counts.
6	The final decision is made.
7	The IFPUG membership is informed of the decision.
8	Changes become effective with, and are reflected in, the next release of the Counting Practices Manual.

Issue Submitted

The reader submits ideas, changes, or issues to the Counting Practices Committee using the Reader's Request Form at the end of this manual. If the page is not available, send comments to the address in the front of the manual and mark it, "ATTN: Counting Practices Committee."

Research Assigned

A member of the CPC is assigned the responsibility for identifying all alternatives, the rationale, and the potential impact of each alternative if it is implemented. Thorough examination of existing counting standards and historical papers is completed while compiling alternatives. In addition, an effort is made to determine what is thought to be *common practice*.

CPC Review

The CPC reviews and discusses the rationale for each alternative, and its potential impact. The review and discussion may result in a proposal for change or the review may lead the committee to reject the change request.

Solution Proposed

A proposed solution is made to the IFPUG membership and written comments are solicited.

A copy of the proposed changes is mailed to IFPUG contacts at member organizations. The proposal also may be announced and distributed during an IFPUG conference. The latter depends on the timing of the committee meeting rather than the conference schedule.

Impact Study Initiated

The CPC has adopted a conservative stance on initiating impact studies. If it is possible that *common practice* must change, or several organizations or types of applications will be impacted by the change, an impact study is initiated.

The success of the impact study is the responsibility of every IFPUG member. If the CPC receives written feedback indicating there is little or no impact, the study is discontinued.

Final Decision Made

The committee makes a final decision using results from research, written comments from members, and the impact study.

The committee can complete more than one iteration of Steps 2 through 5 (research through impact study) before making a final decision. The final decision can result in a change or the committee may decide that a change is not warranted.

Decision Communi-cated

The final decision is communicated in writing to IFPUG members via the IFPUG contact at the various organizations.

If any impact study results contributed to making a decision, the results and a recommendation on how to minimize the impact of the change will also be communicated.

Decision Effective Date

The Counting Practices Manual is updated to reflect the decisions. The effective date of the decisions is the date of the next January release of the manual.

Related IFPUG Documentation

This Counting Practices Manual is one module in the IFPUG documentation. All documents complement each other.

The following table describes the other publications.

Document	Description
IFPUG Brochure (Available)	This publication is an introduction to the International Function Point Users Group. It includes a brief history of the organization, introduces function point analysis, and defines the purpose of IFPUG. The brochure also includes a membership application. Audience: This publication is for anyone who wants an overview of IFPUG or an application for membership.
IFPUG: Organizational Structure and Services (Available)	This publication describes IFPUG services, and lists the board of directors, committees, and affiliate members worldwide. Audience: This publication is for anyone who wants background information about IFPUG.
Guidelines for Software Measurement (Release Date: April 1994; currently under revision by the Management Reporting Committee)	This manual provides an overview of software metrics for organizations working to create or improve software measurement programs. The manual addresses both system and customer management, provides high-level justifications for software measurement, and examines the components of effective measurement programs. Audience: This manual is intended for IFPUG members, Function Point Coordinators, persons who prepare the reports to management, and other persons knowledgeable about and working directly with function points.
Quick Reference Counting Guide (Release Date: January 1999)	This quick reference guide is a summary of function point counting rules and procedures. Audience: This summary information is intended for anyone applying function point analysis.
Function Point Analysis Case Studies (Release Dates: Case Study 1, Release 2: September 2000 Case Study 2 Release 2: April 2001 Case Study 3 Release 2: September 2001 Case Study 4: September 1998); currently under revision	The case studies illustrate the major counting techniques that comprise the Function Point Counting Practices Manual. The cases illustrate function point counts for a sample application. The cases include the counting that occurs at the end of the analysis phase of software development and after system construction. Audience: The case studies are intended for persons new to function point analysis as well as those with intermediate and advanced experience.

Document	Description
IFPUG Glossary (Available with *CPM* and *Guidelines for Software Measurement*)	This is a comprehensive glossary that defines terms used across IFPUG publications. Audience: The glossary is recommended for anyone who receives any of the other IFPUG documents or anyone who needs definitions of IFPUG terms.
"A Framework for Functional Sizing", IFPUG, September 2003	This paper explains that product size contains three dimensions: functional size, technical size and quality size. The IFPUG FPA-method provides a measure for the functional size.
"IT Measurement: Practical Advice from the Experts", Addison-Wesley, April 2002	This book is an excellent compilation of articles written by experts in the field of Information Technology. It was compiled by IFPUG to include recent insights in the application of software metrics in practice.

Training Requirements

Usability evaluations of this publication have verified that reading the Counting Practices Manual alone is not sufficient training to apply function point counting at the optimum level. Training is recommended, particularly for those new to function point counting.

Note: For function point training, be sure you are trained using IFPUG certified materials. Call the IFPUG Executive Office at 609-799-4900 for a list of instructors with certified training courses.

In addition to the function point specific information, this manual includes the use of structured analysis and design terms, such as business systems and entity. The glossary includes definitions of these terms, but the Counting Practices Manual does not include detailed explanations of structured analysis and design techniques. Therefore, all of the material will not apply or be helpful if you have not been trained in structured analysis and design techniques.

Part 1 - Process and Rules

This page intentionally left blank.

1

Part 1 Process and Rules

Introduction

Part 1 provides the function point analysis process for sizing software following the IFPUG Method as well as the detailed rules for identifying and counting data functions and transactional functions.

Part 1 includes:

- An overview of function point analysis and definition of terms
- A description of the objectives and benefits of function point analysis
- An explanation of the counting process and rules and the procedures to be followed in counting
- A definition of the user concept
- Recommendations on when to size during the software life cycle
- Definitions of the types of count
- Identification of the counting scope and application boundaries
- The definitions, rules, procedures, and counting hints for data functions and transactional functions
- Definitions, scoring, hints, and examples for each of the fourteen general system characteristics
- Formulas for calculating function point counts

Contents

Part 1 includes the following chapters:

Topic	See Page
Overview of Function Point Analysis	2-1
User View	3-1
Determine Type of Count	4-1

Topic	See Page
Identify Counting Scope and Application Boundary	5-1
Count Data Functions	6-1
Count Transactional Functions	7-1
Determine Value Adjustment Factor	8-1
Calculate Adjusted Function Point Count	9-1
Index	i-1

2

Overview of Function Point Analysis

Introduction This chapter presents an overview of the function point counting process. It includes the objectives of function point counting and presents a summary and example of the function point counting procedures.

Contents This chapter includes the following sections:

Topic	See Page
Objectives and Benefits of Function Point Analysis	**2-2**
Objectives of Function Point Analysis	2-2
Benefits of Function Point Analysis	2-2
Function Point Counting Procedure	**2-3**
Procedure Diagram	2-3
Procedure by Chapter	2-3
Summary Counting Example	**2-4**
Summary Diagram	2-4
Determine the Type of Function Point Count	2-5
Identify the Counting Scope and Application Boundary	2-5
Determine the Unadjusted Function Point Count	2-6
Determine the Value Adjustment Factor	2-9
Calculate the Adjusted Function Point Count	2-9

Objectives and Benefits of Function Point Analysis

Function point analysis is a standard method for measuring software development from the user's point of view.

Objectives of Function Point Analysis

Function point analysis measures software by quantifying the functionality the software provides to the user based primarily on logical design. With this in mind, the objectives of function point analysis are to:

- Measure functionality that the user requests and receives
- Measure software development and maintenance independently of technology used for implementation

In addition to meeting the above objectives, the process of counting function points should be:

- Simple enough to minimize the overhead of the measurement process
- A consistent measure among various projects and organizations

Benefits of Function Point Analysis

Organizations can apply function point analysis as:

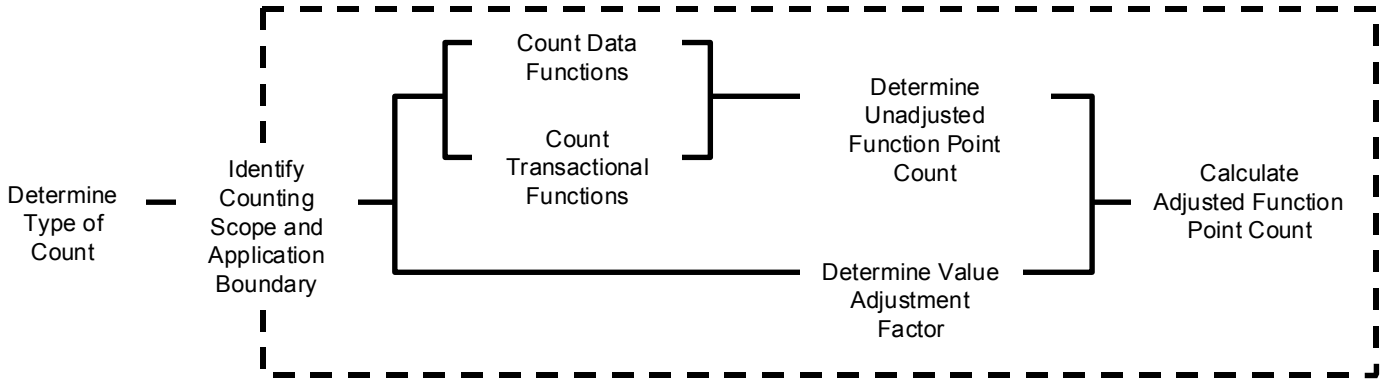
- A tool to determine the size of a purchased application package by counting all the functions included in the package
- A tool to help users determine the benefit of an application package to their organization by counting functions that specifically match their requirements
- A tool to measure the units of a software product to support quality and productivity analysis
- A vehicle to estimate cost and resources required for software development and maintenance
- A normalization factor for software comparison

Refer to other IFPUG documents such as *Function Points as an Asset* for additional information about the benefits of function point analysis, or see the IFPUG web site at http://www.ifpug.org for additional information.

Function Point Counting Procedure

This section presents the high-level procedure for function point counting.

Procedure Diagram



Procedure by Chapter

The following table shows the function point counting procedures as they are explained in the remaining chapters of the manual.

Note: A summary example of the counting procedures is presented on the following pages in this chapter.

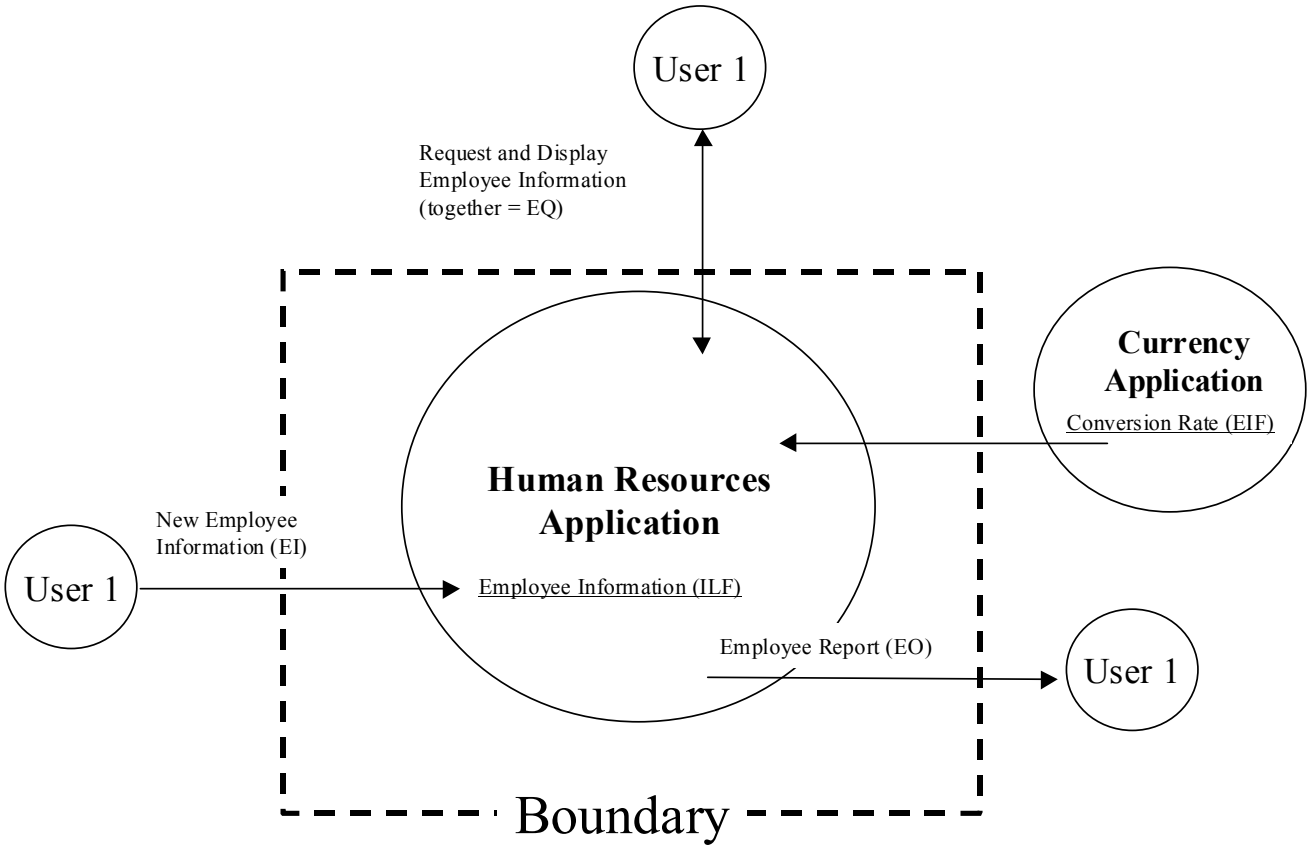
Chapter	Procedure
4	Determine the type of function point count.
5	Identify the counting scope and application boundary.
6	Count the data functions to determine their contribution to the unadjusted function point count.
7	Count the transactional functions to determine their contribution to the unadjusted function point count.
8	Determine the value adjustment factor.
9	Calculate the adjusted function point count.

Summary Counting Example

This section presents a summary example of the function point counting procedure and the components that comprise the count.

Summary Diagram

The following diagram shows the components for the example function point count for a Human Resources Application. Refer to the diagram while reading the remaining paragraphs in this chapter.



Determine the Type of Function Point Count

The first step in the function point counting procedure is to determine the type of function point count.

Function point counts can be associated with either projects or applications. There are three types of function point counts:

- Development project function point count
- Enhancement project function point count
- Application function point count

The example on page 2-4 is for a project function point count, which will also evolve into an application function point count.

Chapter 4 includes detailed definitions of each type of function point count. Chapter 9, the last chapter in Part 1, explains the formulas to calculate the adjusted function point count for each of the three types of counts.

Identify the Counting Scope and Application Boundary

The counting scope defines the functionality which will be included in a particular function point count.

The application boundary indicates the border between the software being measured and the user.

The example on page 2-4 shows the application boundary between the Human Resources Application being measured and the external Currency Application. It also shows the application boundary between the Human Resources Application and the user.

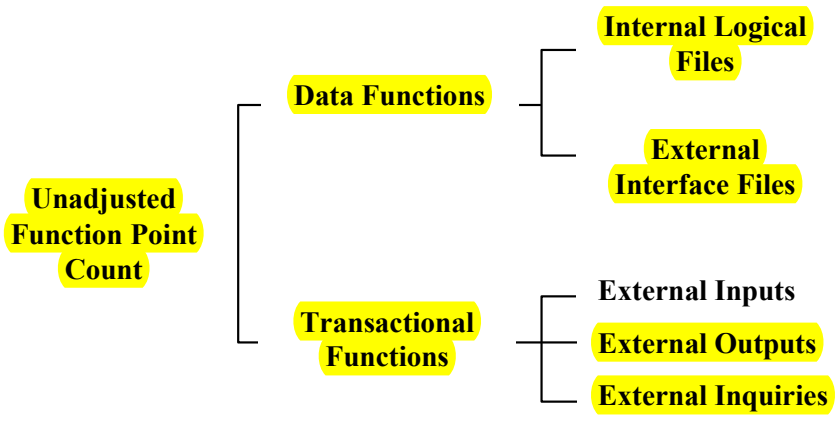
Chapter 5 explains counting scope and application boundary.

Determine the Unadjusted Function Point Count

The unadjusted function point count (UFPC) reflects the specific countable functionality provided to the user by the project or application.

The application's specific user functionality is evaluated in terms of *what* is delivered by the application, not *how* it is delivered. *Only* user-requested and defined components are counted.

The unadjusted function point count has two function types—data and transactional. These function types are further defined as shown in the following diagram.



Count Data Functions

Data functions represent the functionality provided to the user to meet internal and external data requirements. Data functions are either internal logical files or external interface files.

- An internal logical file (ILF) is a user identifiable group of logically related data or control information maintained within the boundary of the application. The primary intent of an ILF is to hold data maintained through one or more elementary processes of the application being counted.

  The example on page 2-4 shows a group of related employee data maintained within the Human Resources Application.

- An external interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted. This means an EIF counted for an application must be in an ILF in another application.

  The example on page 2-4 shows conversion rate information maintained by the Currency Application and referenced by the Human Resources Application.

Chapter 6 of Part 1 explains the rules for counting the data functions.

Part 2 contains procedures for counting data functions.

Part 3 contains examples that illustrate the application of the rules for counting data functions.

Count Transactional Functions

Transactional functions represent the functionality provided to the user to process data. Transactional functions are either external inputs, external outputs, or external inquiries.

- An external input (EI) is an elementary process that processes data or control information that comes from outside the application's boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system.

  The example on page 2-4 shows the process of entering employee information into the Human Resources Application.

- An external output (EO) is an elementary process that sends data or control information outside the application's boundary. The primary intent of an external output is to present information to a user through processing logic other than or in addition to the retrieval of data or control information. The processing logic must contain at least one mathematical formula or calculation, or create derived data. An external output may also maintain one or more ILFs and/or alter the behavior of the system.

  The example on page 2-4 shows the process of producing a report that lists all employees stored in the Human Resources Application.

- An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information. The processing logic contains no mathematical formula or calculation, and creates no derived data. No ILF is maintained during the processing, nor is the behavior of the system altered.

  The example on page 2-4 shows the process of inquiring on employee information (input request) and viewing an employee's information when it appears on a screen (output retrieval).

Chapter 7 of Part 1 explains the rules for counting transactional functions.

Part 2 contains procedures for counting transactional functions.

Part 3 contains examples that illustrate the application of the rules for counting transactional functions.

Determine the Value Adjustment Factor

The value adjustment factor (VAF) indicates the general functionality provided to the user of the application.

The VAF is comprised of 14 general system characteristics (GSCs) that assess the general functionality of the application. Each characteristic has associated descriptions that help determine the degree of influence of the characteristic. The degrees of influence range on a scale of zero to five, from no influence to strong influence.

Chapter 8 explains how to determine the value adjustment factor.

Calculate the Adjusted Function Point Count

The adjusted function point count is calculated using a specific formula for a development project, enhancement project, or application (system baseline) function point count.

Chapter 9 includes formulas and detailed explanations for each of the three types of function point counts.

This page intentionally left blank.

3

User View

Introduction This chapter presents the concept of the user's role in defining the functional requirements for a project or an application.

Contents This chapter includes the following sections:

Topic	See Page
Definition of User View	**3-2**
Sizing During the Life Cycle of an Application	**3-3**
Phase: Initial User Requirements	3-4
Phase: Initial Technical Requirements	3-5
Phase: Final Functional Requirements	3-6
Life Cycle Phase Comparisons	**3-8**
Functional Size	**3-9**
Hints to Help with Counting	**3-10**

Definition of User View

A *user view* represents a formal description of the user's business needs in the user's language. Developers translate the user information into information technology language in order to provide a solution.

A function point count is accomplished using the information in a language that is common to both user(s) and developers.

A user view:

- Is a description of the business functions
- Is approved by the user
- Can be used to count function points
- Can vary in physical form (e.g., catalog of transactions, proposals, requirements document, external specifications, detailed specifications, user handbook)

Sizing During the Life Cycle of an Application

User requirements evolve rapidly in the early phases of a project. Decisions must be agreed upon by the users and the developer on which functions will be included in an application. These decisions regarding the functions of the project may be influenced by:

- The needs of the organization
- The risk (business and technical) associated with the project
- The resources available (e.g. budget, staff) in the organization for the project
- The technology available in the organization
- The influence of either users or developers through comments and suggestions

At the beginning of a project, the feasibility study is produced. The feasibility study is the highest level of specification and is usually very short; for example:

- The organization needs an application to comply with a new tax law
- The organization needs an application to manage inventory more efficiently
- The organization needs an application to manage human resources more efficiently

After the feasibility study, the user develops requirements that become more precise over time. At some point, the user will consult with software developers to create the detailed requirements. Software developers can get an early start with their own development and implementation requirements based upon the feasibility study. The discussions between the user and the software developers lead to enhanced requirements. The development process varies among different organizations. This manual will consider, for illustration purposes, a model with three categories of requirements documents:

- Initial User Requirements
- Initial Technical Requirements
- Final Functional Requirements.

As with other development methodologies, the Final Functional Requirements Phase is the most accurate phase to count function points.

Phase: Initial User Requirements

This phase represents user requirements prior to the sessions between the users and the software developers. It may have one or more of the following characteristics:

- Incomplete

  For example, Initial User Requirements may lack functions necessary for referential integrity.

- Lack "utility" functionality

  For example, essential validation reports or inquiries may be missing.

- Impossible to implement or very difficult to use

  For example, a user may ask for an on-line inquiry that requires an hour of CPU processing.

- Too general

  For example, requirements may not include the number of fields.

- Varying functional needs, if more than one user is responsible for the project

  For example, the requirements of a specific project may vary from one user to another if they do not have the same functional needs.

- Stated requirements without regard for application boundaries

  For example, current and/or future application boundaries may not have been considered.

- Expressed in a different context or a terminology incompatible with function point analysis

  For example, Initial User Requirements may refer to the physical or manual aspects of the system.

Example

In the Human Resources Department of a corporation, a user expresses his requirements as:

"Whenever I'm working with an employee, I want to be able to view the employee's information by entering his or her name."

This requirement implies the development of an inquiry screen and a group of data on employees. (To keep the example simple, assume that the employee group of data is maintained internally by other employee functions, such as create, update and delete employee, which are not described here).

Functions of the Initial User Requirements example**:**

EQ inquiry on a specific employee

ILF employee group of data

Phase: Initial Technical Requirements

This second phase represents the software developers' view of requirements created from the feasibility study. One task of the software developers, among others, is to organize the requirements into existing applications, if any. The Initial Technical Requirements may include elements which are necessary for the implementation, but which are not used in function point counting (e.g., temporary files, index, etc.). This phase may have one or more of the following characteristics:

- Technology dependence

  For example, physical files vary based on the database environment.

- Incorrect identification of the functional needs of the users

  For example, software developers may add functions not requested by the users.

- Terminology unfamiliar to the users

  For example, software developers may refer to physical files rather than to logical groups of data.

- Functionality may be determined by placing too much emphasis on technical constraints

  For example, some developers tend to limit the scope of the requirements by focusing on the computing capacity (CPU) currently available in the organization.

- Boundaries are determined according to the technical architecture of the other applications in the organization

  For example, there may be separate technical requirements for client and server, but they would be contained in the same application boundary when counting function points.

Example Continuing with the same example, the developer states:

"I recognize the need for an employee inquiry. An index is necessary to speed up the retrieval of specific employees."

Functions of the Initial Technical Requirements might be identified as:

EQ inquiry on a specific employee

ILF employee group of data

ILF* index on the employee file

*Index files are not counted. In this example, the index file was incorrectly identified as an ILF to illustrate a potential counting error by software developers.

Phase: Final Functional Requirements

This third phase of requirements results from joint sessions between the user(s) and the software developer(s). The joint sessions are necessary to achieve consistent and complete functional requirements for the application. This phase is the final version of the functional requirements before the development phase begins and has the following characteristics:

- Contains terminology which can be understood by both users and software developers
- Provides integrated descriptions of all user requirements, including requirements from different users
- Is complete and consistent enough to accurately count function points
- Each process and group of data is approved by the user
- The feasibility and usability are approved by the software developers

Example Continuing with the same example:

User: "Whenever I'm working with an employee, I want to be able to view the employee's information by entering his or her name."

Developer: "I recognize the need for an employee inquiry, but many employees may have the same name. It is not possible to specify an individual employee by typing his/her name; therefore, I suggest an on-line employee list (name, location and social security number) from which to select an employee. An index will be necessary to speed up the retrieval of a specific employee."

User: "I agree that the employee selection list is necessary in this case, and it may also be used for purposes other than selecting an employee."

Result of the discussions between the user and the developer:

- Add the on-line list of employees to the functional requirements and the function point count
- Exclude the employee index from the function point count since it is a technical solution

Functions of the Final Functional Requirements example:

EQ	inquiry on a specific employee
EQ	on-line list of employees
ILF	employee group of data

The Final Functional Requirements document is the final version of the requirements before beginning the development phase. At this time, there should be agreement that the documented requirements are complete, formal and approved. The function point count, assuming no additional changes of scope, should be consistent with the count at the completion of development.

Life Cycle Phase Comparisons

Prior to beginning a function point count, determine the application's life cycle phase and whether you are approximating or measuring. Document any assumptions.

Approximating permits assumptions about unknown functions and/or their complexity to accomplish a function point analysis.

Measuring includes the identification of all functions and their complexity to accomplish a function point analysis.

At an early stage, Initial Users Requirements could be the only document available for function point analysis. Despite the disadvantages, this count can be very useful to produce an early estimate. Uses of function point analysis for approximating at the various life cycle phases is presented below:

Life Cycle Phase	Size can be approximated	Size can be measured
Proposal: users express needs and intentions	yes	no
Requirements: developers and users review and agree upon expression of user needs and intentions	yes	yes
Design: developers may include elements for implementation that are not used for function point analysis	yes	yes
Construction	yes	yes
Delivery	yes	yes
Corrective Maintenance	yes	yes

Note: No specific development life cycle is implied. If using an iterative approach, you may expect to approximate for some time into the application life cycle.

Be aware and measure only new or refined agreed upon user needs and intentions.

Functional Size

The IFPUG CPM has been transformed into an ISO standard for functional size measurement with the exclusion of the General System Characteristics (which measure technical requirements). To some extent, this transformation has allowed the Counting Practices Committee to address issues such as code tables consistently. The most important consideration in relation to these issues is how the technical requirements affect the size. Since they are not part of the functional size, they do not contribute to the unadjusted function point count. However, they are still part of the requirements, and therefore contribute to the requirements size.

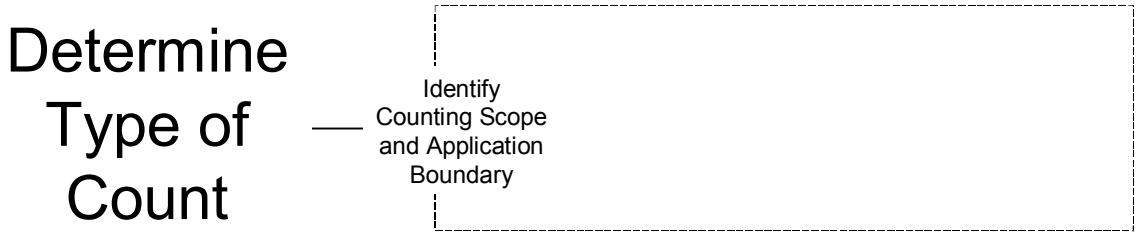
For a detailed discussion on functional size and how it will both guide and constrain the evolution of function point analysis, refer to the IFPUG Framework for Functional Sizing[1].

[1] Framework for Functional Sizing, IFPUG CPC, Release 1.0, September 2003

Hints to Help with Counting

The following hints may help identify the user view of an application and apply function point analysis.

- Do not assume that one physical file equates to one logical file when viewing data logically from the user perspective.
- Although some storage technologies, such as tables in a relational DBMS or a sequential flat file, relate closely to ILFs or EIFs, do not assume that this is always equal to a one-to-one physical-logical relationship.
- Do not assume all physical files must be counted or included as part of an ILF or EIF.
- Look at the different paper forms currently used by the user(s) when identifying transactional functions.
- A transaction which occurs in multiple physical inputs, transaction files or screens, but which has identical processing logic typically corresponds to one transactional function (EI, EO, EQ).
- Remember that one physical report, screen or batch output file can, when viewed logically, correspond to a number of EOs/EQs.
- Remember that two or more physical reports, screens or batch output files can correspond to one EO/EQ if the processing logic is identical.
- Remember that resorting or rearranging a set of data does not make processing logic unique.



Introduction

The first step of the function point counting procedure is to identify the type of function point count.

This chapter includes a detailed explanation of the types of function point counts: development project, enhancement project, and application.

Contents

This chapter includes the following sections:

Topic	See Page
Definitions: Types of Function Point Counts	**4-2**
Development Project	**4-2**
Enhancement Project	4-2
Application	4-2
Diagram of Types of Counts	**4-3**
Estimated and Final Counts	4-3

Definitions: Types of Function Point Counts

Function point counts can be associated with either projects or applications. There are three types of function point counts:

- Development project
- Enhancement project
- Application

The following paragraphs define each type of function point count.

Note: Chapter 9 includes the formulas used to calculate the adjusted function point count for each of the three types of counts.

Development Project

The development project function point count measures the functions provided to the users with the first installation of the software delivered when the project is complete.

Enhancement Project

The enhancement project function point count measures the modifications to the existing application that add, change, or delete user functions delivered when the project is complete.

When the functionality from an enhancement project is installed, the application function point count must be updated to reflect changes in the application's functionality.

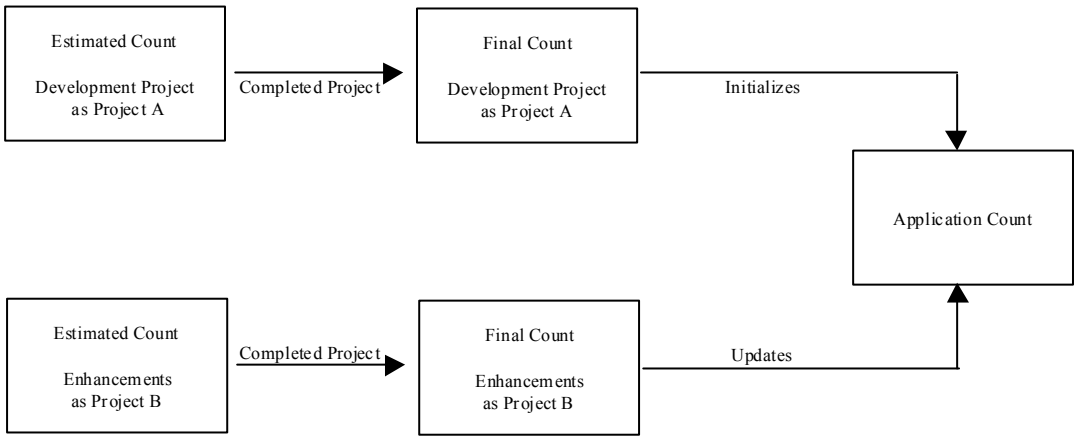
Additional guidelines are included in Part 2.

Application

The application function point count is associated with an installed application. It is also referred to as the *baseline* or *installed* function point count. This count provides a measure of the current functions the application provides the user. This number is initialized when the development project function point count is completed. It is updated every time completion of an enhancement project alters the application's functions.

Diagram of Types of Counts

The following diagram illustrates the types of function point counts and their relationships. (Project A is completed first, followed by Project B.)

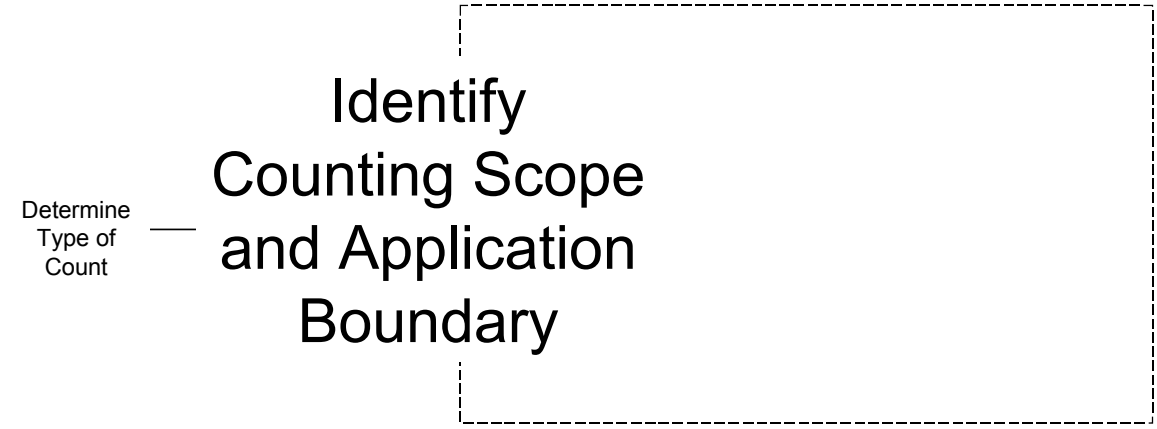


Estimated and Final Counts

It is important to realize that early function point counts are estimates of delivered functionality. In addition, as the scope is clarified and the functions developed, it is quite normal to identify additional functionality that was not specified in the original requirements. This phenomenon is sometimes called *scope creep*.

It is essential to update application counts upon completion of the project. If the functionality changes during development, the function point count at the end of the life cycle should accurately reflect the full functionality delivered to the user.

This page intentionally left blank.



Introduction

This chapter defines the terms: purpose of the count, counting scope and application boundary. It includes rules, procedures, and hints to determine boundaries for applications and to establish the scope of the count.

Contents

This chapter includes the following sections:

Topic	See Page
Definition of Counting Scope and Application Boundary	**5-2**
Definition of the Purpose of the Count	5-2
Definition of the Counting Scope	5-3
Definition of the Application Boundary	5-4
Counting Scope and Application Boundary Rules and Procedures	**5-5**
Boundary Rules	5-5
Counting Scope and Application Boundary Procedures	5-5
Hints to Help to Identify the Counting Scope and the Application Boundary	**5-6**

Definition of Counting Scope and Application Boundary

This section defines counting scope and application boundary and explains how they are influenced by the purpose of the count.

Definition of the Purpose of the Count

The purpose of a function point count is to provide an answer to a business problem.

The purpose:

- Determines the type of function point count and the scope of the required count to obtain the answer to the business problem under investigation
- Influences the positioning of the boundary between the software under review and the surrounding software; e.g., if the Personnel Module from the Human Resources System is to be replaced by a package, the users may decide to reposition the boundary and consider the Personnel Module as a separate application

Examples of purposes are:

- To provide a function point count as an input to the estimation process to determine the effort to develop the first release of an application
- To provide a function point count of the installed base of applications
- To provide a function point count to enable the comparison of functionality delivered by two different suppliers' packages

Definition of the Counting Scope

The counting scope defines the functionality which will be included in a particular function point count.

The scope:

- Defines a (sub) set of the software being sized
- Is determined by the purpose for performing the function point count
- Identifies which functions will be included in the function point count so as to provide answers relevant to the purpose for counting
- Could include more than one application

The *scope* of:

- An enhancement function point count includes all the functions being added, changed and deleted. The boundary of the application(s) impacted remains the same. The functionality of the application(s) reflects the impact of the functions being added, changed or deleted.
- A development function point count includes all functions impacted (built or customized) by the project activities.
- An application function point count may include, depending on the purpose (e.g., provide a package as the software solution):
  - only the functions being used by the user
  - all the functions delivered

  The application boundary of the two counts is the same and is independent of the *scope*.

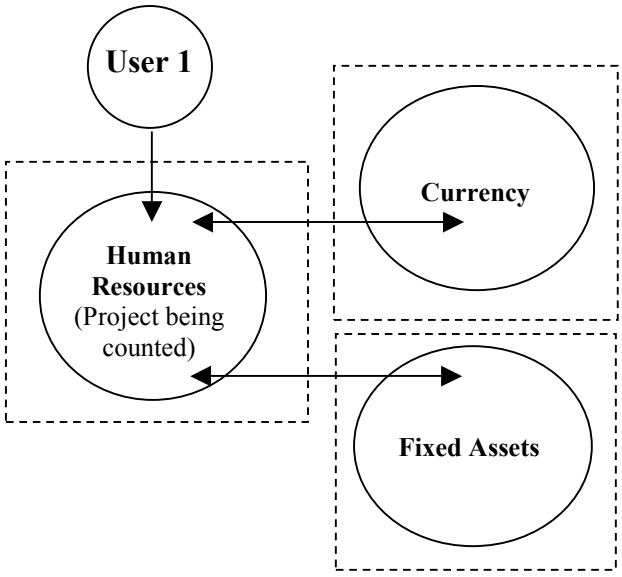
Definition of the Application Boundary

The application boundary indicates the border between the software being measured and the user.

The application boundary:

- Defines what is external to the application
- Is the conceptual interface between the 'internal' application and the 'external' user world
- Acts as a 'membrane' through which data processed by transactions (EIs, EOs and EQs) pass into and out from the application
- Encloses the logical data maintained by the application (ILFs)
- Assists in identifying the logical data referenced by but not maintained within this application (EIFs)
- Is dependent on the user's external business view of the application. It is independent of technical and/or implementation considerations

For example, the following diagram shows boundaries between the Human Resources application and the external applications, Currency and Fixed Assets. The example also shows the boundary between the human user (User 1) and the Human Resources application.



Counting Scope and Application Boundary Rules and Procedures

This section defines the rules and procedures that apply when identifying counting scope and application boundaries.

The position of the application boundary is important because it impacts the result of the function point count. The application boundary assists in identifying the data entering the application which will be included in the scope of the count.

Boundary Rules

The following rules must apply for boundaries:

- The boundary is determined based on the user's view. The focus is on what the user can understand and describe.
- The boundary between related applications is based on separate functional areas as seen by the user, not on technical considerations.
- The initial boundary already established for the application or applications being modified is not influenced by the counting scope.

Note: There may be more than one application included in the counting scope. If so, multiple application boundaries would be identified.

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

Counting Scope and Application Boundary Procedures

When you perform a function point count, the following characteristics of the count should be properly documented:

Step	Action
1	Establish the purpose of the count
2	Identify the counting scope
3	Identify the application boundary
4	Document the following items: • The purpose of the count • The counting scope • The application boundary • Any assumptions related to the above

Hints to Help to Identify the Counting Scope and the Application Boundary

Counting Scope

The following hints can help you to identify the counting scope:

- Review the purpose of the function point count to help determine the counting scope.
- When identifying the scope of the installed base function point count (i.e., the functionality supported by the maintenance team), include all functions currently in production and used by the users.

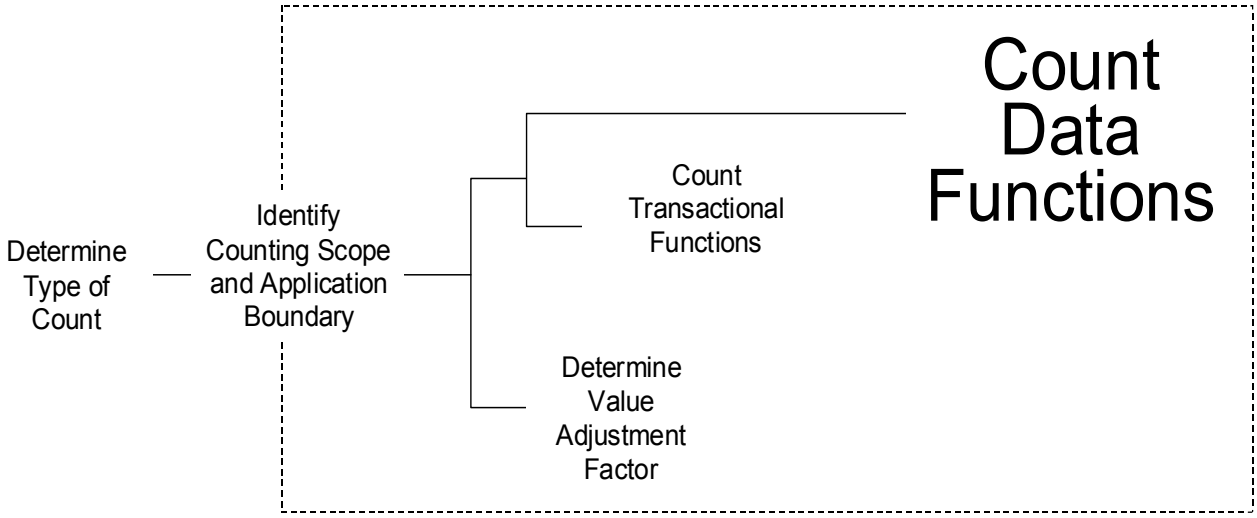
Application Boundary

The following hints can help you to identify the application boundary:

- Use the system external specifications or get a system flow chart and draw a boundary around it to highlight which parts are internal and which are external to the application.
- Look at how groups of data are being maintained.
- Identify functional areas by assigning ownership of certain types of analysis objects (such as entities or elementary processes) to a functional area.
- Look at associated measurement data, such as effort, cost, and defects. The boundaries for function points and the other measurement data should be the same.

Hints

The positioning of the application boundary between the software under investigation and other software applications may be subjective. It is often difficult to delineate where one application stops and another begins. Try to place the boundary from a business perspective rather than based on technical or physical considerations. It is important that the application boundary is placed with care, since all data crossing the boundary can potentially be included in the scope of the count.



Introduction

Data functions represent the functionality provided to the user to meet internal and external data requirements. Data function types are defined as internal logical files (ILFs) and external interface files (EIFs).

The term *file* here does not mean file in the traditional data processing sense. In this case, file refers to a logically related group of data and not the physical implementation of those groups of data.

This chapter includes the definitions for internal logical files and external interface files and explains the counting procedures and rules associated with these functions.

Contents

This chapter includes the following sections:

Topic	See Page
Definitions: ILFs and EIFs	**6-3**
Internal Logical Files	6-3
External Interface Files	6-3
Difference between ILFs and EIFs	6-3
Definitions for Embedded Terms	6-3

Continued on next page

Topic	See Page
ILF/EIF Counting Rules	**6-5**
Summary of Counting Procedures	6-5
ILF Identification Rules	6-6
EIF Identification Rules	6-6
Complexity and Contribution Definitions and Rules	**6-7**
DET Definition	6-7
DET Rules	6-7
RET Definition	6-9
RET Rules	6-9
ILF/EIF Counting Procedures	**6-10**
Procedure Diagram	6-10
Identification Procedures	6-10
Complexity and Contribution Procedures	6-11
Hints to Help with Counting	**6-13**

Definitions: ILFs and EIFs

This section includes the definitions of the internal logical files (ILFs) and external interface files (EIFs). Embedded terms within the definitions are defined and examples are included throughout this definition section.

Internal Logical Files

An internal logical file (ILF) is a user identifiable group of logically related data or control information maintained within the boundary of the application. The primary intent of an ILF is to hold data maintained through one or more elementary processes of the application being counted.

External Interface Files

An external interface file (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application counted. This means an EIF counted for an application must be in an ILF in another application.

Difference between ILFs and EIFs

The primary difference between an internal logical file and an external interface file is that an EIF **is not** maintained by the application being counted, while an ILF is.

Definitions for Embedded Terms

The following paragraphs further define ILFs and EIFs by defining embedded terms within the definitions.

Control Information

Control Information is data that influences an elementary process of the application being counted. It specifies what, when, or how data is to be processed.

For example, someone in the payroll department establishes payment cycles to schedule when the employees for each location are to be paid. The payment cycle, or schedule, contains timing information that affects when the elementary process of paying employees occurs.

User Identifiable

The term *user identifiable* refers to defined requirements for processes and/or groups of data that are agreed upon, and understood by, both the user(s) and software developer(s).

For example, users and software developers agree that a Human Resources Application will maintain and store Employee information in the application.

Maintained

The term *maintained* is the ability to modify data through an elementary process.

Examples include, but are not limited to, add, change, delete, populate, revise, update, assign, and create.

Elementary Process

An *elementary process* is the smallest unit of activity that is meaningful to the user(s).

For example, a user requires the ability to add a new employee to the application. The user definition of employee includes salary and dependent information. From the user perspective, the smallest unit of activity is to add a new employee. Adding one of the pieces of information, such as salary or dependent, is not an activity that would qualify as an elementary process.

The *elementary process* must be self-contained and leave the business of the application being counted in a consistent state.

For example, the user requirements to add an employee include setting up salary and dependent information. If all the employee information is not added, an employee has not yet been created. Adding some of the information alone leaves the business of adding an employee in an inconsistent state. If both the employee salary and dependent information is added, this unit of activity is completed and the business is left in a consistent state.

ILF/EIF Counting Rules

This section defines the rules that apply when counting internal logical files and external interface files.

Summary of Counting Procedures

This summary is included to show the rules in the context of the ILF and EIF counting procedures.

Note: The detailed counting procedures begin on page 6-10.

The ILF and EIF counting procedures include the following two activities:

Step	Action
1	Identify the ILFs and EIFs.
2	Determine the ILF or EIF complexity and their contribution to the unadjusted function point count.

ILF and EIF counting rules are used for each activity. There are two types of rules:

- Identification rules
- Complexity and contribution rules

The following list outlines how the rules are presented:

- ILF identification rules
- EIF identification rules
- Complexity and contribution rules, which include:
  - Data element types (DETs)
  - Record element types (RETs)

ILF Identification Rules

To identify ILFs, look for groups of data or control information that satisfy the definition of an ILF.

All of the following counting rules must apply for the information to be counted as an ILF.

- The group of data or control information is logical and user identifiable.
- The group of data is maintained through an elementary process within the application boundary being counted.

EIF Identification Rules

To identify EIFs, look for groups of data or control information that satisfy the definition of an EIF.

All of the following counting rules must apply for the information to be counted as an EIF.

- The group of data or control information is logical and user identifiable.
- The group of data is referenced by, and external to, the application being counted.
- The group of data **is not maintained** by the application being counted.
- The group of data is maintained in an ILF of another application.

Complexity and Contribution Definitions and Rules

The number of ILFs, EIFs, and their relative functional complexity determine the contribution of the data functions to the unadjusted function point count.

Assign each identified ILF and EIF a functional complexity based on the number of data element types (DETs) and record element types (RETs) associated with the ILF or EIF.

This section defines DETs and RETs and includes the counting rules for each.

DET Definition

A *data element type* is a unique user recognizable, non-repeated field.

DET Rules

The following rules apply when counting DETs:

- Count a DET for each unique user recognizable, non-repeated field maintained in or retrieved from the ILF or EIF through the execution of an elementary process.

  For example, an account number that is stored in multiple fields is counted as one DET.

  For example, a before or after image for a group of 10 fields maintained for audit purposes would count as one DET for the before image (all 10 fields) and as one DET for the after image (all 10 fields) for a total of 2 DETs.

  For example, the result(s) of a calculation from an elementary process, such as calculated sales tax value for a customer order maintained on an ILF is counted as one DET on the customer order ILF.

  For example, accessing the price of an item which is saved to a billing file or fields such as a time stamp if required by the user(s) are counted as DETs.

  For example, if an employee number which appears twice in an ILF or EIF as (1) the key of the employee record and (2) a foreign key in the dependent record, count the DET only once.

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

- When two applications maintain and/or reference the same ILF/EIF, but each maintains/references separate DETs, count only the DETs being used by each application to size the ILF/EIF.

  For example, Application A may specifically identify and use an address as: street address, city, state and zip code. Application B may see the address as one block of data without regard to individual components.

Application A would count four DETs; Application B would count one DET.

For example, Application X maintains and/or references an ILF that contains a SSN, Name, Street Name, Mail Stop, City, State, and Zip. Application Z maintains and/or references the Name, City, and State. Application X would count seven DETs; Application Z would count three DETs.

- Count a DET for each piece of data required by the user to establish a relationship with another ILF or EIF.

  For example, in an HR application, an employee's information is maintained on an ILF. The employee's job name is included as part of the employee's information. This DET is counted because it is required to relate an employee to a job that exists in the organization. This type of data element is referred to as a *foreign key*.

  For example, in an object oriented (OO) application, the user requires an association between object classes, which have been identified as separate ILFs. Location name is a DET in the Location EIF. The location name is required when processing employee information; consequently, it is also counted as a DET within the Employee ILF.

RET Definition

A *record element type* (RET) is a user recognizable subgroup of data elements within an ILF or EIF.

There are two types of subgroups:

- Optional
- Mandatory

Optional subgroups are those that the user has the option of using one or none of the subgroups during an elementary process that adds or creates an instance of the data.

Mandatory subgroups are subgroups where the user must use at least one.

For example, in a Human Resources Application, information for an employee is added by entering some general information. In addition to the general information, the employee is a salaried or hourly employee.

The user has determined that an employee must be either salaried or hourly. Each type of employee has unique attributes. Either type can have information about dependents. For this example, there are three subgroups or RETs as shown below:

- Salaried employee (mandatory); includes general information
- Hourly employee (mandatory); includes general information
- Dependent (optional)

RET Rules

One of the following rules applies when counting RETs:

- Count a RET for each optional or mandatory subgroup of the ILF or EIF.

Or

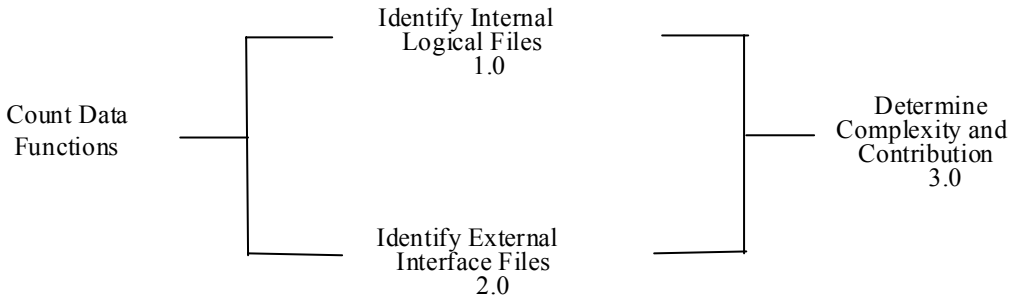
- If there are no subgroups, count the ILF or EIF as one RET.

ILF/EIF Counting Procedures

This section includes detailed explanations of ILF and EIF counting procedures.

Procedure Diagram

The following diagram shows the high-level procedure for counting ILFs and EIFs.



The following paragraphs explain the steps for each activity.

Identification Procedures

Follow these steps to identify ILFs and EIFs:

Step	Action	Rule Set(s) to Use	See Page
1.0	Identify Internal Logical Files	ILF Identification Rules	6-6
2.0	Identify External Interface Files	EIF Identification Rules	6-6
3.0	Determine Complexity and Contribution	Complexity and Contribution Procedures	6-11

Complexity and Contribution Procedures

Follow these steps to calculate ILF and EIF complexity and contribution to the unadjusted function point count.

Step	Action
1	Use the complexity and contribution counting rules that begin on page 6-7 to identify and count the number of RETs and DETs.
2	Rate the functional complexity using the following complexity matrix.

	1 to 19 DET	20 to 50 DET	51 or more DET
1 RET	Low	Low	Average
2 to 5 RET	Low	Average	High
6 or more RET	Average	High	High

3 Translate the ILFs and EIFs to unadjusted function points using the appropriate translation table for either ILFs or EIFs.

ILF Translation Table: Use the following table to translate the ILFs to unadjusted function points.

Functional Complexity Rating	Unadjusted Function Points
Low	7
Average	10
High	15

EIF Translation Table: Use the following table to translate the EIFs to unadjusted function points.

Functional Complexity Rating	Unadjusted Function Points
Low	5
Average	7
High	10

For example, a high complexity rating for an EIF translates to 10 unadjusted function points.

Continued on next page

Step	Action
4	Calculate each ILF and EIF contribution to the unadjusted function point count. For example, the following table shows the calculation for one high complexity ILF, two average and one high complexity EIFs.

Function Type	Functional Complexity			Complexity Totals	Function Type Totals
ILF	0	Low	X 7 =	0	
	0	Average	X 10 =	0	
	1	High	X 15 =	15	15
EIF	0	Low	X 5 =	0	
	2	Average	X 7 =	14	
	1	High	X 10 =	10	24

In this simple example, there are no ILFs of low or average complexity; therefore, the total count for ILFs is 15. For the EIFs, there are no low complexity, 2 average complexity EIFs (14 points) and 1 high complexity (10 points) for an EIF total of 24.
The contributions for ILFs and EIFs will be added to the table that lists all function types. The final total for all function types is the unadjusted function point count. The Appendix includes a table to record the totals for all functions.

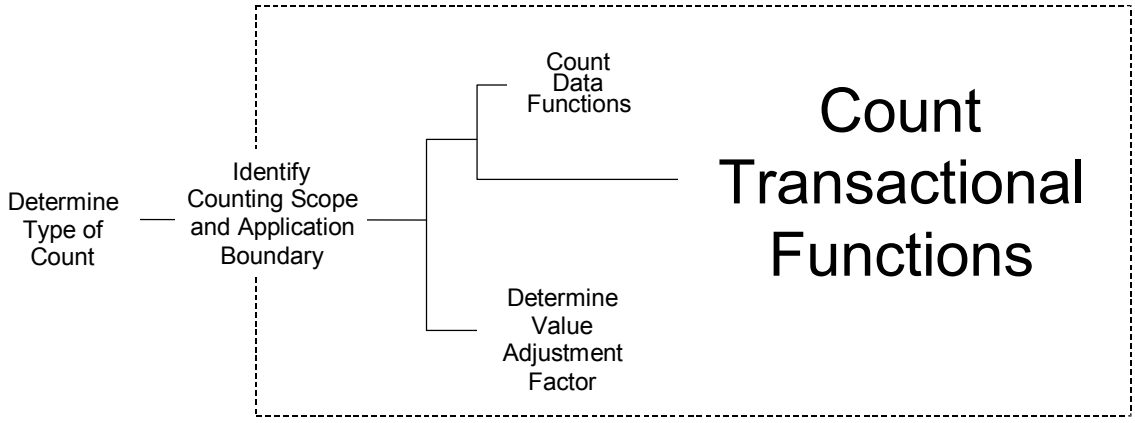
Hints to Help with Counting

The following hints may help you apply the ILF and EIF counting rules.

Caution: These hints *are not* rules and should not be used as rules.

- Is the data a logical group that supports specific user requirements?
  - An application can use an ILF or EIF in multiple processes, but the ILF or EIF is counted only once.
  - A logical file cannot be counted as both an ILF and EIF for the same application. If the data group satisfies both rules, count as an ILF.
  - If a group of data was not counted as an ILF or EIF itself, count its data elements as DETs for the ILF or EIF, which includes that group of data.
  - Do not assume that one physical file, table or object class equals one logical file when viewing data logically from the user perspective.
  - Although some storage technologies such as tables in a relational DBMS or sequential flat file or object classes relate closely to ILFs or EIFs, do not assume that this always equals a one-to-one physical-logical relationship.
  - Do not assume all physical files must be counted or included as part of an ILF or EIF.
- Where is data maintained? Inside or outside the application boundary?
  - Look at the workflow.
  - In the process functional decomposition, identify where interfaces occur with the user and other applications.
  - Work through the process diagram to get hints.
  - Credit ILFs maintained by more than one application to each application at the time the application is counted. Only the DETs being used by each application being counted should be used to size the ILF/EIF.
- Is the data in an ILF maintained through an elementary process of the application?
  - An application can use an ILF or EIF multiple times, but you count the ILF or EIF only once.
  - An elementary process can maintain more than one ILF.
  - Work through the process diagram to get hints.
  - Credit ILFs maintained by more than one application to each application at the time the application is counted.

This page intentionally left blank.



Introduction

Transactional functions represent the functionality provided to the user for the processing of data by an application. Transactional functions are defined as external inputs (EIs), external outputs (EOs), and external inquiries (EQs).

This chapter defines EI, EO, and EQ transactional functions and includes the associated function point counting rules and procedures. The chapter concludes with detailed examples for each function.

Contents

This chapter includes the following sections:

Topic	See Page
Definitions: EIs, EOs and EQs	**7-3**
External Inputs	7-3
External Outputs	7-3
External Inquiry	7-3
Summary of the Functions Performed by EIs, EOs, and EQs	7-4
Definitions for Embedded Terms	7-5
Summary of Processing Logic Used by EIs, EOs and EQs	7-8
EI/EO/EQ Counting Rules	**7-9**
Summary of Counting Procedures	7-9

Continued on next page

Topic	See Page
Elementary Process Identification Rules	7-10
Transactional Functions Counting Rules	7-11
Primary Intent Description for EIs	7-11
External Input Counting Rules	7-11
Primary Intent Description for EOs and EQs	7-12
Shared EO and EQ Counting Rules	7-12
Additional External Output Counting Rules	7-12
Additional External Inquiry Counting Rules	7-13
Complexity and Contribution Definitions and Rules	7-13
FTR Definition	7-13
DET Definition	7-13
EI Complexity and Contribution Rules	7-14
FTR Rules for an EI	7-14
DET Rules for an EI	7-14
EO/EQ Complexity and Contribution Rules	7-16
Shared FTR Rules for EOs and EQs	7-16
Additional FTR Rules for an EO	7-16
Shared DET Rules for EOs and EQs	7-16
EI, EO and EQ Counting Procedures	**7-18**
Procedure Diagram	7-18
Identification Procedures	7-19
Complexity and Contribution Procedures	7-21
Hints to Help with Counting EIs, EOs and EQs	**7-24**
Additional Hints to Help Counting EOs and EQs	7-26

Definitions: EIs, EOs and EQs

This section includes the definitions of EIs, EOs and EQs. Embedded terms within the definitions are defined, and examples are included throughout this definition section.

External Inputs

An external input (EI) is an elementary process that processes data or control information that comes from outside the application boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system.

External Outputs

An external output (EO) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external output is to present information to a user through processing logic other than, or in addition to, the retrieval of data or control information . The processing logic must contain at least one mathematical formula or calculation, create derived data maintain one or more ILFs or alter the behavior of the system.

External Inquiry

An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information from an ILF of EIF. The processing logic contains no mathematical formulas or calculations, and creates no derived data. No ILF is maintained during the processing, nor is the behavior of the system altered.

Summary of the Functions Performed by EIs, EOs, and EQs

The main difference between the transactional function types is their primary intent. The table below summarizes functions that may be performed by each transactional function type, and specifies the primary intent of each. Note the primary intent for an EI—this is the main difference from EOs and EQs. Some of the differences between EOs and EQs are that an EO may perform the functions of altering the behavior of the system or maintaining one or more ILFs when performing the primary intent of presenting information to the user. Other differences are identified in the section below that summarizes forms of processing logic used by each transactional function.

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

Legend:

PI	the primary intent of the transactional function type
F	a function of the transactional function type, but is not the primary intent and is sometimes present
N/A	the function is not allowed by the transactional function type

Definitions for Embedded Terms

The following paragraphs further define EIs, EOs and EQs by defining terms used within the above definitions.

Elementary Process

An *elementary process* is the smallest unit of activity that is meaningful to the user(s).

For example, a user requires the ability to add a new employee to the application. The user definition of employee includes salary and dependent information. From the user perspective, the smallest unit of activity is to add a new employee. Adding one of the pieces of information, such as salary or dependent, is not an activity that would qualify as an elementary process.

The *elementary process* must be self-contained and leave the business of the application being counted in a consistent state.

For example, the user requirements to add an employee include setting up salary and dependent's information. If all the employee information is not added, an employee has not yet been created. Adding some of the information alone leaves the business of adding an employee in an inconsistent state. If both the employee salary and dependent information is added, this unit of activity is completed and the business is left in a consistent state.

Control Information

Control Information is data that influences an elementary process of the application being counted. *It specifies what, when, or how data is to be processed.*

For example, someone in the payroll department establishes payment cycles to schedule when the employees for each location are to be paid. The payment cycle, or schedule, contains timing information that affects when the elementary process of paying employees occurs.

Maintained

The term *maintained* is the ability to modify data through an elementary process.

Examples include, but are not limited to, add, change, delete, populate, revise, update, assign, and create.

User

A *user* is any person that specifies Functional User Requirements and/or any person or thing that communicates or interacts with the software at any time.

Examples include people within the HR department who interact with the application to set up employees, and the Benefits application that interacts with the HR application to receive information about employees' dependents.

Processing Logic

Processing logic is defined as requirements specifically requested by the user to complete an elementary process. Those requirements may include the following actions:

1. Validations are performed

   For example, when adding a new employee to an organization, the employee process has processing logic that validates the information being added.

2. Mathematical formulas and calculations are performed

   For example, when reporting on all employees within an organization the process includes calculating the total number of salaried employees, hourly employees and all employees.

3. Equivalent values are converted

   For example, an elementary process references currency conversion rates from US dollars to other currencies. The conversion is accomplished by retrieving values from tables, so calculations need not be performed.

4. Data is filtered and selected by using specified criteria to compare multiple sets of data

   For example, to generate a list of employees by assignment, an elementary process compares the job number of a job assignment to select and lists the appropriate employees with that assignment.

5. Conditions are analyzed to determine which are applicable

   For example, processing logic exercised by the elementary process when an employee is added and will depend on whether an employee is paid based on salary or hours worked.

6. One or more ILFs are updated

   For example, when adding an employee, the elementary process updates the employee ILF to maintain the employee data.

7. One or more ILFs or EIFs are referenced

   For example, when adding an employee, the currency EIF is referenced to use the correct US dollar conversion rate to determine an employee's hourly rate.

8. Data or control information is retrieved

   For example, to view a list of possible pay grades, pay grade information is retrieved.

9. Derived data is created by transforming existing data to create additional data

   For example, to determine (derive) a patient's registration number (e.g., SMIJO01), the following data is concatenated:

   - the first three letters of the patient's last name (e.g., SMI for Smith)
   - the first two letter of the patient's first name (e.g., JO for John)
   - a unique two-digit sequence number (starting with 01)

10. Behavior of the system is altered

    For example, the behavior of the elementary process of paying employees is altered when a change is made to pay them every other Friday versus on the 15th and the last day of the month.

11. Prepare and present information outside the boundary

    For example, a list of employees displayed for the user.

12. Capability exists to accept data or control information that enters the application boundary

    For example, a user enters several pieces of information to add a customer order to the system.

13. Data is resorted or rearranged

    For example, a user requests the list of employees in alphabetical order.

    Note: Resorting or rearranging a set of data does not impact the identification of the type or uniqueness of a transactional function.

One elementary process may include multiple alternatives or occurrences of the above actions.

For example: validations, filters, resorts, etc.

Summary of Processing Logic Used by EIs, EOs and EQs

The following table summarizes which forms of processing logic may be performed by EIs, EOs, and EQs. For each transactional function type, certain types of processing logic must be performed to accomplish the primary intent of that type. The 13 actions do not by themselves identify unique elementary processes.

Form of Processing Logic:	Transactional Functional Type:		
	EI	EO	EQ
1. Validations are performed	c	c	c
2. Mathematical formula and calculations are performed	c	m*	n
3. Equivalent values are converted	c	c	c
4. Data is filtered and selected by using specified criteria to compare multiple sets of data	c	c	c
5. Conditions are analyzed to determine which are applicable	c	c	c
6. At least one ILF is updated	m*	m*	n
7. At least one ILF or EIF is referenced	c	c	m
8. Data or control information is retrieved	c	c	m
9. Derived data is created	c	m*	n
10. Behavior of the system is altered	m*	m*	n
11. Prepare and present information outside the boundary	c	m	m
12. Capability to accept data or control information that enters the application boundary.	m	c	c
13. Resorting or rearranging a set of data	c	c	c

Legend:

m it is **mandatory** that the function type perform the form of processing logic

m* it is **mandatory** that the function type perform at least one of these (m*) forms of processing logic

c the function type **can** perform the form of processing logic, but it is not mandatory

n function type **cannot** perform the form of processing logic

EI/EO/EQ Counting Rules

This section defines the rules that apply when counting EIs, EOs and EQs.

Summary of Counting Procedures

This summary is included to show the rules in the context of the EI, EO, and EQ counting procedures.

Note: The detailed procedures begin on page 7-18.

The EI, EO and EQ counting procedures include the following steps:

Step	Action
1	Identify the elementary processes.
2	Determine the primary intent of the identified elementary processes, and classify as an EI, EO, or EQ.
3	Validate against the transaction (EI, EO, EQ) identification rules.
4	Determine the transaction (EI, EO, EQ) complexity.
5	Determine the transaction (EI, EO, EQ) contribution to the unadjusted function point count.

The rules are explained in the following paragraphs.

Elementary Process Identification Rules

To identify elementary processes, look for user activities occurring in the application.

All of the following counting rules must apply for the process to be identified as an elementary process.

- The process is the smallest unit of activity that is meaningful to the user.
- The process is self-contained and leaves the business of the application in a consistent state.

Transactional Functions Counting Rules

To classify each elementary process, determine which of the primary intent descriptions apply, and use the associated rules to identify a specific transactional function type.

Primary Intent Description for EIs

The primary intent of an elementary process is to maintain an ILF or alter the behavior of the system.

External Input Counting Rules

For each elementary process that has a primary intent to maintain one or more ILFs or to alter the behavior of the system, apply the following rules to determine if the function should be classified as an external input. All of the rules must apply for the elementary process to be counted as a unique occurrence of an external input.

- The data or control information is received from outside the application boundary.
- At least one ILF is maintained if the data entering the boundary is not control information that alters the behavior of the system.
- For the identified process, one of the following three statements must apply:
  - Processing logic is unique from the processing logic performed by other external inputs for the application.
  - The set of data elements identified is different from the sets identified for other external inputs for the application.
  - The ILFs or EIFs referenced are different from the files referenced by other external inputs in the application.

Primary Intent Description for EOs and EQs

The primary intent of the elementary process is to present information to a user.

Shared EO and EQ Counting Rules

For each elementary process that has a primary intent to present information to a user, apply the following rules to determine if the process may be classified as an external output or external inquiry. All of the rules must apply for the elementary process to be counted as a unique occurrence of an external output or external inquiry.

- The function sends data or control information external to the application boundary.
- For the identified process, one of the following three statements must apply:
  - Processing logic is unique from the processing logic performed by other external outputs or external inquiries for the application.
  - The set of data elements identified is different from the sets identified for other external outputs and external inquiries in the application.
  - The ILFs or EIFs referenced are different from the files referenced by other external outputs and external inquiries in the application.

Additional External Output Counting Rules

In addition to adhering to all shared EO and EQ rules, one of the following rules must apply for the elementary process to be counted as a unique external output.

- The processing logic of the elementary process contains at least one mathematical formula or calculation.
- The processing logic of the elementary process creates derived data.
- The processing logic of the elementary process maintains at least one ILF.
- The processing logic of the elementary process alters the behavior of the system.

Additional External Inquiry Counting Rules

In addition to adhering to all shared EO and EQ rules, all of the following rules must apply for the elementary process to be counted as a unique external inquiry.

- The processing logic of the elementary process retrieves data or control information from an ILF or EIF.
- The processing logic of the elementary process does not contain a mathematical formula or calculation.
- The processing logic of the elementary process does not create derived data.
- The processing logic of the elementary process does not maintain an ILF.
- The processing logic of the elementary process does not alter the behavior of the system.

Complexity and Contribution Definitions and Rules

The number of EIs, EOs, and EQs and their relative functional complexities determine the contribution of the transactional functions to the unadjusted function point count.

Assign each identified EI, EO and EQ a functional complexity based on the number of file types referenced (FTRs) and data element types (DETs).

FTR Definition

A *file type referenced* is

- An internal logical file read or maintained by a transactional function or
- An external interface file read by a transactional function

DET Definition

A *data element type* is a unique user recognizable, non-repeated field.

EI Complexity and Contribution Rules

This section defines FTR and DET rules used to determine the complexity and contribution of external inputs.

FTR Rules for an EI

The following rules apply when counting FTRs:

- Count an FTR for each ILF maintained.
- Count an FTR for each ILF or EIF read during the processing of the external input.
- Count only one FTR for each ILF that is both maintained and read.

DET Rules for an EI

The following rules apply when counting DETs:

- Count one DET for each user recognizable, non-repeated field that enters or exits the application boundary and is required to complete the external input.

  For example, job name and pay grade are two fields that the user provides when adding a job.

- Do not count fields that are retrieved or derived by the system and stored on an ILF during the elementary process if the fields did not cross the application boundary.

  For example, when the customer order is added to the system, the unit price is automatically retrieved for each ordered item and stored on the billing record. The unit price would not be counted as a DET for the EI because it did not cross the boundary when the user adds the customer order.

  For example, in order to maintain the US hourly rate for hourly employees working in other countries with other currencies, the local hourly rate is provided by the user. During the processing of all the pieces of data provided to add an employee, a conversion rate is retrieved from the currency system to calculate the US hourly rate. The calculated US hourly rate is maintained on the employee ILF as a result of adding the employee. The US hourly rate would not be counted as a DET for the EI because it does not enter the boundary, but is internally calculated (i.e., it is derived data).

- Count one DET for the capability to send a system response message outside the application boundary to indicate an error occurred during processing, confirm that processing is complete or verify that processing should continue.

  For example, if a user tries to add an existing employee to a Human Resources application, the system generates one of several error messages and the incorrect field is highlighted. Count one DET that includes all the system responses which indicate the error conditions, confirm that processing is complete or verify that processing should continue.

- Count one DET for the ability to specify an action to be taken even if there are multiple methods for invoking the same logical process.

  For example, if the user can initiate the adding of an employee clicking on the OK button or by pressing a PF key, count one DET for the ability to initiate the process.

EO/EQ Complexity and Contribution Rules

This section defines FTR and DET rules used to determine the complexity and contribution of external outputs and external inquiries.

Shared FTR Rules for EOs and EQs

The following rule applies when counting FTRs for both EOs and EQs:

- Count one FTR for each ILF or EIF read during the processing of the elementary process.

Additional FTR Rules for an EO

The following additional rules apply when counting FTRs for EOs:

- Count one FTR for each ILF maintained during the processing of the elementary process.
- Count only one FTR for each ILF that is both maintained and read during the elementary process.

Shared DET Rules for EOs and EQs

The following rules apply when counting DETs for both EOs and EQs.

- Count one DET for each user recognizable, non-repeated field that enters the application boundary and is required to specify when, what and/or how the data is to be retrieved or generated by the elementary process.

  For example (EO/EQ), to generate a list of employees, employee name is a field the user provides when indicating which employees to list.

- Count one DET for each user recognizable, non-repeated field that exits the boundary.

  For example (EO/EQ), a text message may be a single word, sentence, or phrase—a line or paragraph included on a report to indicate an explanatory comment counts as a single DET.

  For example (EO/EQ), an account number or date physically stored in multiple fields is counted as one DET when it is required as a single piece of information.

  For example (EO/EQ), a pie chart might have a category label and a numerical equivalent in a graphical output. Count two DETs —one for designating the category and one for the numerical value.

- If a DET both enters and exits the boundary, count it only once for the elementary process.
- Count one DET for the capability to send a system response message outside the application boundary to indicate an error occurred during processing, confirm that processing is complete or verify that processing should continue.

  For example (EO/EQ), if a user tries to request a listing, but does not have access to the information, count one DET for the system response.
- Count one DET for the ability to specify an action to be taken even if there are multiple methods for invoking the same logical process.

  For example (EO/EQ), if the user can initiate the generation of a report by clicking on the OK button or by pressing a PF key, count one DET for the ability to initiate the report.
- Do not count fields that are retrieved or derived by the system and stored on an ILF during the elementary process if the fields did not cross the application boundary.

  For example (EO), when a paycheck is printed, a status field on the employee ILF is updated to indicate that the check has been printed. Do not count the status field as a DET since it did not cross the boundary.
- Do not count literals as DETs.

  For example (EO/EQ), literals include report titles, screen or panel identification, column headings, and field titles.
- Do not count paging variables or system-generated stamps.

  For example (EO/EQ), system-generated variables and stamps include

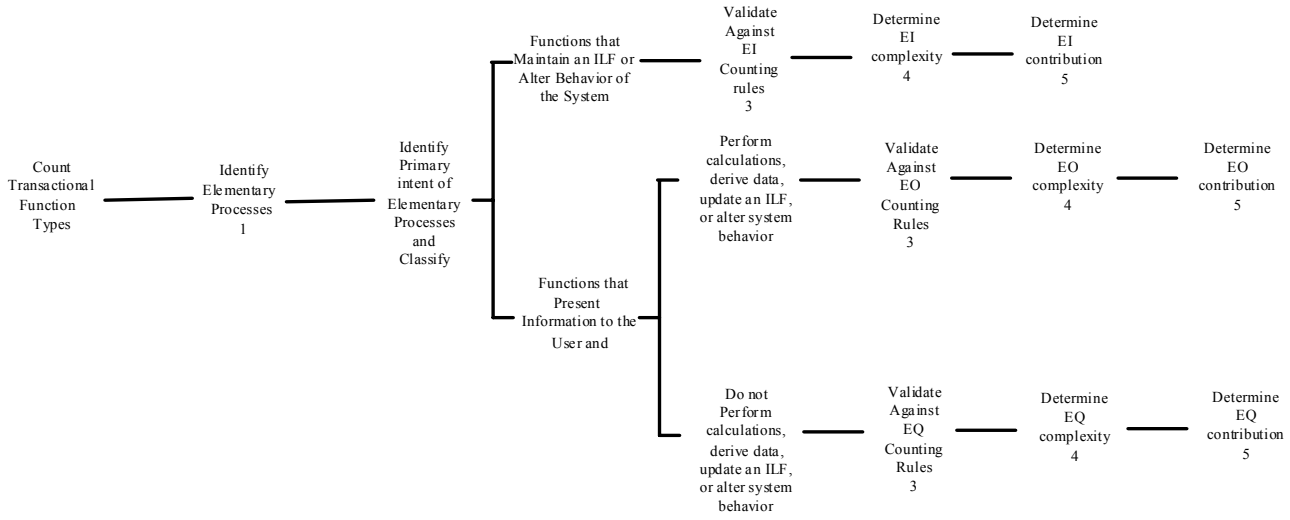
  - Page numbers
  - Positioning information such as "Rows 37 to 54 of 211"
  - Paging commands such as previous, next, and paging arrows on a GUI application
  - Date and time fields if they are displayed.

EI, EO and EQ Counting Procedures

This section includes detailed explanations of EI, EO and EQ counting procedures.

Procedure Diagram

The following diagram shows the high-level procedure for counting EIs, EOs and EQs:



The following paragraphs explain the steps for each activity.

Identification Procedures

Follow these steps to identify EIs, EOs and EQs:

Step	Action	Rule Set(s) to Use	Page #
1	Identify Elementary Processes	Elementary Process Identification Rules	7-10
2	Identify Primary Intent of Elementary Processes, and classify as an EI, EO, or EQ.	Transactional Function Type Counting Rules - Primary Intent Descriptions for:	
		• EIs	7-11
		• EOs and EQs	7-12
For the Elementary Processes where the Primary Intent is to maintain an ILF or to alter the behavior of the system:			
3	Validate against the EI identification rules.	Transactional Function Type Counting Rules: • External Input Counting Rules	7-11
4	Determine EI Complexity	Refer to Complexity and Contribution Procedures	7-21
5	Determine EI Contribution	Refer to Complexity and Contribution Procedures	7-23
For the Elementary Processes where the Primary Intent is to present information to the user and that perform calculations, derive data, update an ILF, or alter the behavior of the system.			
3	Validate against the EO identification rules.	Transactional Function Type Counting Rules:	
		• Shared EO and EQ Counting Rules	7-12
		• Additional EO Counting Rules	7-12

Continued on next page

Step	Action	Rule Set(s) to Use	Page #
4	Determine EO Complexity	Refer to Complexity and Contribution Procedures	7-22
5	Determine EO Contribution	Refer to Complexity and Contribution Procedures:	7-23
For the Elementary Processes where the Primary Intent is to present information to the user and that do not perform calculations, derive data, update an ILF, or alter the behavior of the system.			
3	Validate against the EQ identification rules.	Transactional Function Type Counting Rules: • Shared EO and EQ Counting Rules • Additional EQ Counting Rules	 7-12 7-13
4	Determine EQ Complexity	Refer to Complexity and Contribution Procedures:	7-22
5	Determine EQ Contribution	Refer toComplexity and Contribution Procedures	7-23

Complexity and Contribution Procedures

Follow these steps to calculate EI, EO and EQ complexity and contribution to the unadjusted function point count:

Step	Complexity Procedure
4	**External Inputs:** Use the Complexity Definitions and Rules for EIs that begin on page 7-14 to identify and count the number of FTRs and DETs. Rate the complexity of the EI using the following complexity matrix.

	1 to 4 DET	5 to 15 DET	16 or more DET
0 to 1 FTR	Low	Low	Average
2 FTRs	Low	Average	High
3 or more FTRs	Average	High	High

Step	Complexity Procedure
4	**External Outputs and External Inquiries:** Use the Complexity Definitions and Rules for EOs or EQs that begin on page 7-16 to identify and count the number of FTRs and DETs. Rate the complexity of the EOs or EQs using the following complexity matrix. Remember to use the cumulative number of FTRs and DETs, ignoring duplicates, to rate the complexity.

	1 to 5 DET	6 to 19 DET	20 or more DET
0 to 1 FTR	Low	Low	Average
2 to 3 FTRs	Low	Average	High
4 or more FTRs	Average	High	High

Step	Contribution Procedure
5	**External Inputs and External Inquiries:** Use the following table to translate the EI or EQ complexity to unadjusted function points.

Functional Complexity Rating	Unadjusted Function Points
Low	3
Average	4
High	6

Step	Contribution Procedure
5	**External Outputs:** Use the following table to translate the EO to unadjusted function points.

Functional Complexity Rating	Unadjusted Function Points
Low	4
Average	5
High	7

Hints to Help with Counting EIs, EOs and EQs

The following hints may help you apply the EI, EO and EQ counting rules.

Caution: The hints *are not* rules and should not be used as rules.

- Is data received from outside the application boundary?
  - Look at the work flow.
  - Identify where the user and other application interfaces occur in the process functional decomposition.
- Is the process the smallest unit of activity from the user perspective?
  - Look at the different paper or on-line forms used.
  - Review the ILFs to identify how the user groups the information.
  - Identify where the user and other application interfaces occur in the process functional decomposition.
  - Look at what happened in the manual system.
  - Note that one physical input or transaction file or screen can, when viewed logically, correspond to a number of EIs, EOs or EQs.
  - Note that two or more physical input or transaction files or screens can correspond to one EI, EO or EQ if the processing logic is identical.
- Is the process self-contained and does it leave the business in a consistent state?
  - Review other external inputs, external outputs and external inquiries to understand how the user works with the information.
  - Work through the process diagram to get hints.
  - Look at what happened in the manual system.
  - Check for consistency with other decisions.
- Is the processing logic unique from other EIs, EOs and EQs?
  - Identify batch inputs or outputs based on the processing logic required.
  - Remember that sorting or rearranging a set of data does not make processing logic unique.
- Are the data elements different from those for other EIs, EOs or EQs?
  - If the data elements appear to be a subset of the data elements of another EI, EO, or EQ, be sure two elementary processes are required by the user—one for the main data elements and one for the subsets.

- Identify the primary intent of the elementary process before classifying it as an EI, EO, or EQ.
- Identification of the elementary process(es) is based on a joint understanding or interpretation of the requirements between the user and the developers.
- Each element in a functional decomposition may not map to a unique elementary process.
- The identification of the elementary processes requires interpretation of the user requirements.
- Count only one FTR for each ILF/EIF referenced even if the ILF/EIF has multiple RETs.

Additional Hints to Help Counting EOs and EQs

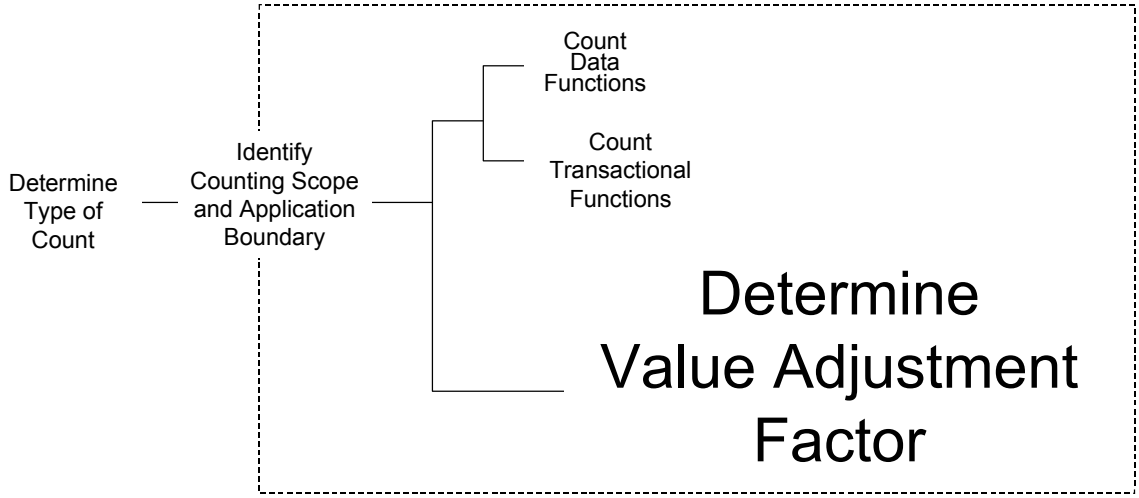
- Is the process the smallest unit of activity from the user perspective?
  - An EO or EQ can be triggered by a process inside the application boundary.

    For example, the user requires that a report of all changed employee pay rates be sent to the budgeting area every 8 hours based on an internal clock.

    Situation A. The report contains employee name, SSN, and hourly pay rate which are all retrieved from the employee file. This is the smallest unit of activity from the user's perspective, contains no mathematical formulas or calculations, and no ILF is maintained in the process. This is one EQ.

    Situation B. The report contains employee name, SSN, and hourly pay rate which are all retrieved from the employee file. The report also includes the percentage pay change for the employee which is calculated from the data on the employee file. This is the smallest unit of activity from the user's perspective, and no ILF is maintained in the process However, since the process contains a mathematical formula, this is one EO.
  - Derived data for an EO does not have to be displayed on the output.

    For example, each month, a report is generated listing all employees due for appraisal in the next 30 days. The records are selected by calculating next appraisal date based on the employee's last appraisal date, which is a field on the employee file, and the current date plus 30 days. This would be counted as one EO, and not as an EQ.



Introduction This chapter explains the value adjustment factor for the function point count.

Contents This chapter includes the following sections:

Topic	See Page
Value Adjustment Factor Determination	8-3
Procedures to Determine the VAF	8-3
General System Characteristics	8-4
Degrees of Influence	8-5
1. Data Communications	8-6
2. Distributed Data Processing	8-9
3. Performance	8-11
4. Heavily Used Configuration	8-13
5. Transaction Rate	8-15
6. Online Data Entry	8-16
7. End-User Efficiency	8-17
8. Online Update	8-19

Continued on next page

Topic	See Page
9. Complex Processing	8-20
10. Reusability	8-22
11. Installation Ease	8-23
12. Operational Ease	8-25
13. Multiple Sites	8-27
14. Facilitate Change	8-29

Value Adjustment Factor Determination

The value adjustment factor (VAF) is based on 14 general system characteristics (GSCs) that rate the general functionality of the application being counted. Each characteristic has associated descriptions that help determine the degree of influence of that characteristic. The degree of influence for each characteristic ranges on a scale of zero to five, from no influence to strong influence.

The 14 general system characteristics are summarized into the value adjustment factor. When applied, the value adjustment factor adjusts the unadjusted function point count +/-35 percent to produce the adjusted function point count.

Procedures to Determine the VAF

The following steps outline the procedures to determine the value adjustment factor.

Step	Action
1	Evaluate each of the 14 general system characteristics on a scale from zero to five to determine the degree of influence (DI).
2	Add the degrees of influence for all 14 general system characteristics to produce the total degree of influence (TDI).
3	Insert the TDI into the following equation to produce the value adjustment factor. VAF = (TDI * 0.01) + 0.65 For example, the following value adjustment factor is calculated if there are three degrees of influence for each of the 14 GSC descriptions (3*14). VAF = (42 * 0.01) + 0.65 VAF = 1.07

A table to facilitate the calculation is included in the Appendix.

General System Characteristics

The *general system characteristics* are a set of 14 questions that evaluate the overall complexity of the application.

The 14 general system characteristics are:

1. Data Communications
2. Distributed Data Processing
3. Performance
4. Heavily Used Configuration
5. Transaction Rate
6. Online Data Entry
7. End-User Efficiency
8. Online Update
9. Complex Processing
10. Reusability
11. Installation Ease
12. Operational Ease
13. Multiple Sites
14. Facilitate Change

Degrees of Influence

Based on the stated user requirements, each general system characteristic (GSC) must be evaluated in terms of its degree of influence (DI) on a scale of zero to five.

Score as	System Influence
0	Not present or no influence
1	Incidental influence
2	Moderate influence
3	Average influence
4	Significant influence
5	Strong influence throughout

Guidelines to Determine Degree of Influence

Each of the following general system characteristic descriptions includes guidelines to determine the degree of influence.

Each guideline contains a definition of the GSC, rules for determining the score, and, in situations where the rule needs further clarification, hints have been provided to help apply the rules consistently across all platforms.

Hints are not intended to cover all situations but are meant to provide additional guidance in determining the appropriate score.

1. Data Communications

Definition

Data Communications describes the degree to which the application communicates directly with the processor.

The *data* and *control* information used in the application are sent or received over communication facilities. Devices connected locally to the control unit are considered to use communication facilities. Protocol is a set of conventions that permit the transfer or exchange of information between two systems or devices. All data communication links require some type of protocol.

Score

Score As	Descriptions to Determine Degree of Influence
0	Application is pure batch processing **or** a stand-alone application
1	Application is batch but has remote data entry ***or*** remote printing
2	Application is batch but has remote data entry ***and*** remote printing
3	Application includes on-line data collection or TP (teleprocessing) front end to a batch process or query system
4	Application is more than a front-end, but supports ***only one*** type of TP communications
5	Application is more than a front-end, and supports ***more than one*** type of TP communications protocol

Hints

Protocol examples include FTP, dial-up, Token Ring, Ethernet, SNA, TCP/IP, IPX/SPX, HTTP, XML, WAP, NTP, ICQ, and NETBEUI. This list should ***not*** be considered exhaustive.

Hints to Rules 1 and 2

- Remote devices might include 3270 terminal connected to a mainframe computer that allows only simple edits (numeric vs. alpha), or printers
connected via parallel port (the user can specify where to direct the
output).
- The entry of data does not involve reading or writing directly to an ILF. Data are entered on-line, but the transactions are stored in a temporary file for batch update of ILF(s) at a later time.
- The entry of data does not involve reading or writing directly to an ILF.

Hints to Rule 3

- Simple business rules and minimal edits (e.g., alpha/numeric, range check, required data, etc.) may be performed. When this data is eventually processed by the application, additional edits are performed.
- The entry of data does not involve reading or writing directly to an ILF. Data are entered on-line, but the transactions are stored in a temporary file for batch update of ILF(s) at a later time.

Hints to Rule 4

- Data for the application is collected and may directly update ILF(s) or be stored for future processing using an input device, which performs edits based on business rules.
- Only one communication protocol is used. Typically, when this data is processed by the application, no further edits are required.
- The entry of data involves reading or writing to an ILF.
- For example, client-server data entry or Internet data entry, but not both.

Hints to Rule 5

- Same as **4,** however, data collection is performed using multiple telecommunication protocols.
- For example, client-server data entry and Internet data entry of the same transaction.

Typically

- Batch applications receive a score of 0 to 3
- On-line applications receive a score of 4
- Web-based applications receive a score of 4 or 5
- Real-time, telecommunication, or process control systems receive a score of 4 or 5

2. Distributed Data Processing

Definition

Distributed Data Processing describes the degree to which the application transfers data among physical components of the application.

Distributed data or processing functions are a characteristic of the application within the application boundary.

Score

Score As	Descriptions To Determine Degree of Influence
0	Data is not transferred or processed on another component of the system.
1	Data is prepared for transfer, then is transferred and processed on another component of the system, for user processing.
2	Data is prepared for transfer, then is transferred and processed on another component of the system ,***not*** for user processing.
3	Distributed processing and data transfer are on-line and in **one** direction only.
4	Distributed processing and data transfer are on-line and in **both** directions
5	Distributed processing and data transfer are on-line and are dynamically performed on the most appropriate component of the system.

Hints

Distributed data processing by definition is not an application that is contained on a central processor, which sends data to other applications. In a distributed environment, the application is viewed as requiring multiple components (hardware) on which certain processing or data resides. A knowledgeable user would usually recognize this configuration.

Hints to Rule 0

- Presentation, processing, and I/O components are all in the same place (i.e., stand-alone applications).

Hints to Rule 1

- Application downloads data to a user's client machine, so the user can use Excel or other reporting tools to prepare graphs and perform other analysis.
- Process that transfers data from mainframe to an external component for user processing. This transfer is performed using a simple protocol such as FTP.
- Transferred to a user for processing.

Hints to Rule 2

- Process that transfers data from mainframe to mid-tier. For example;, processing with SAS-PC.
- Application sends data to client or server. This data is then processed or used to produce reports, etc. No data or confirmation is sent back to the client or server.
- Transferred to a component for processing.

Hints to Rule 3

- Data is sent between client and server in **one** direction only. This data is then processed or used to produce reports, etc. by the receiving application. This data typically includes transactions that update an ILF on the client or server.
- For example – client-server or web-enabled applications.

Hints to Rule 4

- Data is sent between client and server in **either** direction. This data is then processed or used to produce reports, etc. by the receiving application. This data typically includes transactions that update an ILF on the client or server.
- For example - client-server or web-enabled applications.
- The application runs under an operating system that automatically handles the allocation between components, **however,** the use of the operating system did not influence the design and implementation of the application.

Hints to Rule 5

- The developer must consider special application software that looks at multiple processors and runs the application on a specific type of processor. This is invisible to the user.
- The application runs under an operating system that automatically handles the dynamic allocation between components, **and** the use of the operating system specifically influenced the design and implementation of the application.

Typically

- Most applications, including legacy applications, receive a score of 0
- Primitive distributed applications which that include batch applications in which data is not transferred online on-line receive a score of 1 to 2
- Client-server or web-based applications receive a score of 3 to 4
- It is uncommon to score 5
- There must be multiple servers or processors, each of which would be selected dynamically on the basis of its real-time availability to score 5

3. Performance

Definition

Performance describes the degree to which response time and throughput performance considerations influenced the application development.

Application performance objectives, stated or approved (or implied) by the user, *in either* response or throughput, influence (or will influence) the design, development, installation, and support of the application.

Score

Score As	Descriptions To Determine Degree of Influence
0	No special performance requirements were stated by the user..
1	Performance and design requirements were stated and reviewed but no special actions were required.
2	Response time or throughput is critical during ***peak*** hours. No special design for CPU utilization was required. Processing deadline is for the next business cycle.
3	Response time or throughput is critical during ***all business*** hours. No special design for CPU utilization was required. Processing deadline requirements with interfacing systems are constraining.
4	In addition, stated user performance requirements are stringent enough to require performance analysis tasks in the design phase.
5	In addition, performance analysis tools were used in the design, development, and/or implementation phases to meet the stated user performance requirements.

Hints

- GSCs 3, 4 and 5 are somewhat related. For this GSC, think in terms of "How fast can we make the application go and how much did/does that impact the design, development, and/or implementation?"
- The users may require real time access to their data, stating or implying standards for response time and throughput capacity.
- Response time typically relates to interactive processing; throughput relates to batch processing

Typically

- Batch applications receive a score of 0 to 4
- On-line (including interactive client-server or web-enabled) applications receive a score of 0 to 4
- Web-based applications receive a score of 4 or 5
- Most MIS on-line systems receive a score of 2
- Real-time, telecommunication, or process control systems receive a score of 0 to 5
- A score of 5 requires the use of performance analysis tools

4. Heavily Used Configuration

Definition

Heavily Used Configuration describes the degree to which computer resource restrictions influenced the development of the application.

A heavily used operational configuration may require special considerations when designing the application. For example, the user wants to run the application on existing or committed equipment that will be heavily used.

Score

Score As	Descriptions To Determine Degree of Influence
0	No explicit or implicit operational restrictions are included.
1	Operational restrictions do exist, but are less restrictive than a typical application. No special effort is needed to meet the restrictions.
2	Operational restrictions do exist, but are typical for an application. Special effort through controllers or control programs is needed to meet the restrictions.
3	Stated operational restrictions require special constraints on ***one*** piece of the application in the central processor or a dedicated processor.
4	Stated operational restrictions require special constraints on the ***entire*** application in the central processor or a dedicated processor.
5	In addition, there are special constraints on the application in the distributed components of the system.

Hints

- GSCs 3, 4 and 5 are somewhat related.
- For this GSC think in terms of "How much does the infrastructure influence the design?"

Examples

Examples of operational restrictions may include the following (not an exhaustive list):

- This question indicates that the application must run on a computer that is under- powered and can not adequately handle the new or changed functionality and that somehow the developers can overcome this by developing the application differently.
- More than one application accessing the same data can create operational restrictions.
- Application competing for the same resource and technologies with the potential deadlocks must be tuned and constrained to avoid performance degradation.

Typically

- Most applications receive a score of 2
- Client-server, web-enabled, real-time, telecommunication or process control systems receive a score of 3 to 5, but then you would need either a dedicated processor or multiple processors processing the same transactions and searching for the most expeditious means of processing.

5. Transaction Rate

Definition

Transaction Rate describes the degree to which the rate of business transactions influenced the development of the application.

The transaction rate is high, and it influences the design, development, installation, and support of the application. Users may require what they regard as normal response time even during times of peak volume.

Score

Score As	Descriptions To Determine Degree of Influence
0	No peak transaction period is anticipated.
1	Low transaction rates have minimal effect on the design, development, and installation phases
2	Average transaction rates have some effect on the design, development, and installation phases.
3	High transaction rates affect the design, development, and/or installation phases.
4	High transaction rate(s) stated by the user in the application requirements or service level agreements are high enough to require performance analysis tasks in the design, development, and/or installation phases.
5	High transaction rate(s) stated by the user in the application requirements or service level agreements are high enough to require performance analysis tasks and, in addition, require the use of performance analysis tools in the design, development, and/or installation phases.

Hints

- GSCs 3, 4 and 5 are somewhat related. For this GSC think in terms of "How many transactions can be processed by the application in a given period of time?"
- Often this score is the same as the score as the score for GSC 3 because transaction rates often influence performance requirements.

Typically

- Batch applications receive a score of 0 to 3
- On-line (including interactive client-server or web-enabled) applications receive a score of 0 to 4
- Real-time, telecommunication, or process control systems receive a score of 0 to 5
- A score of 5 requires the use of performance analysis tools

6. Online Data Entry

Definition

On-line User Interface describes the degree to which data is entered or retrieved through interactive transactions.

On-line User Interface for data entry, control functions, reports, and queries are provided in the application.

Score

Score As	Descriptions To Determine Degree of Influence
0	All transactions are processed in batch mode.
1	1% to 7% of transactions are interactive.
2	8% to 15% of transactions are interactive.
3	16% to 23% of transactions are interactive.
4	24% to 30% of transactions are interactive.
5	More than 30% of transactions are interactive.

Hints

- This refers to types of transactions *not* volumes.
- For example, if an application has 45 EIs, EOs, and EQs, what percent of the EIs, EOs, and EQs are accomplished via on-line transactions.

Typically

- Batch applications receive a score of 0 to 1
- On-line, real-time, telecommunication, or process control systems receive a score of 5
- Most contemporary on-line (including interactive client-server or web-enabled) applications receive a score of 5
- Batch systems with on-line feature may have a lot of batch transactions, but is must be at least 71 percent batch to receive a score of less than 5.

7. End-User Efficiency

Definition

User Efficiency describes the degree of consideration for human factors and ease of use for the user of the application measured.

The on-line functions provided emphasize a design for user efficiency (human factor/user friendliness). The design includes:

- Navigational aids (e.g., function keys, jumps, dynamically generated menus, hyper-links)
- Menus
- On-line help and documents
- Automated cursor movement
- Scrolling
- Remote printing (via on-line transmissions)
- Pre-assigned function keys (e.g., clear screen, request help, clone screen)
- Batch jobs submitted from on-line transactions
- Drop down List box
- Heavy use of reverse video, highlighting, colors, underlining, and other indicators
- Hard-copy documentation of on-line transactions (e.g., screen print)
- Mouse interface
- Pop-up windows
- Templates and/or defaults
- Bilingual support (supports two languages: count as four items)
- Multi-lingual support (supports more than two languages: count as six items)

Score

Score As	Descriptions To Determine Degree of Influence
0	None of the above.
1	One to three of the above.
2	Four to five of the above.
3	Six or more of the above, but there are no specific user requirements related to efficiency.
4	Six or more of the above, and stated requirements for user efficiency are strong enough to require **design tasks** for human factors to be included
5	Six or more of the above, and stated requirements for user efficiency are strong enough to require **use of special tools and processes** in order to demonstrate that the objectives have been achieved.

Hints

- Use a convention of a score of 4 whenever the application is deployed in a GUI environment (unless it scores 5).
- Usually only software environments that prepare applications for mass-market or non-technical users score 5, and only if they have ergonomics specialists and/or usability studies as part of their process.

Typically

- Pure batch applications receive a score of 0
- Character mode user interface receive a score of 1 or possibly a 2
- GUI user interface to be used for low volume transactions receive a score of 3
- GUI user interface to be used for high volume transactions and most Web ***Intranet*** user interfaces receive a score of 4 (requires design tasks for human factors)
- Web ***Internet*** user interfaces receive a score of 5 (requires special tools and processes to demonstrate that the objectives have been achieved)

8. Online Update

Definition

On-line Update describes the degree to which internal logical files are updated on-line.

The application provides on-line update for the internal logical files.

Score

Score As	Descriptions To Determine Degree of Influence
0	None.
1	On-line update of one to three control files is included. Volume of updating is low and recovery is easy.
2	On-line update of four or more control files is included. Volume of updating is low and recovery is easy.
3	On-line update of major internal logical files is included.
4	In addition, protection against data loss is essential and has been specially designed and programmed in the system.
5	In addition, high volumes bring cost considerations into the recovery process. Highly automated recovery procedures with minimum human intervention are included.

Hints

- On-line update usually requires a keyed file or database.
- Automatic recovery provided by the operating system counts if it impacts the application.

Typically

- Pure batch applications receive a score of 0.
- On-line updates of files that modify the way an application processes or validates data receive a score of 1 or 2.
- On-line updates of user persistent data receive a score of 3.
- MIS applications receive a score of 3 or less.
- Most GUI type applications receive a score of 3 or above.
- Applications which use programmed recovery such as SQL roll back and commit receive a score of 4. Operational/routine backup is not considered protection against data loss.
- Applications required to recover data, reboot, or perform other self-contained functions in the event of a system error receive a score of 5. Recovery may require a human to press enter or perform some other minimal function to initiate this process.

9. Complex Processing

Definition

Complex processing describes the degree to which processing logic influenced the development of the application. The following components are present:

- Sensitive control and/or application-specific security processing.
- Extensive logical processing
- Extensive mathematical processing
- Much exception processing, resulting in incomplete transactions that must be processed again.
- Complex processing to handle multiple input/output possibilities.

Score

Score As	Descriptions To Determine Degree of Influence
0	None of the above.
1	Any one of the above.
2	Any two of the above.
3	Any three of the above.
4	Any four of the above.
5	All five of the above.

Hints

- Sensitive control or security process (e.g., individual users would have different access authority to screens where they could view and/or change data) may include special audit processing (audit data would be captured whenever data was viewed and/or changed and reported).
- Application-specific security processing may include internally developed security processing or use of purchased security packages.
- **Extensive logical** processing is Boolean logic (use of 'AND', 'OR') of greater than average difficulty or a minimum of 4 nested conditional (IF, CASE) statements. Extensive logical processing does not occur in most MIS applications.
- **Extensive mathematical** processing is arithmetic that is beyond the capability of a 4-function calculator (add, subtract, multiply, divide). This is usually not present in most MIS applications. However, an engineering application may qualify.

Hints

- Exception processing includes incomplete ATM transactions caused by TP interruption, missing data values, failed validations, or cycle redundancy checks which can be used to recreate lost pieces of data
- Multiple input/output possibilities include multi-media, device independence, voice, OCR reading, barcode reading, retinal scanning, and Breathalyzer analysis.

Typically

Scoring is not platform dependent.

10. Reusability

Definition

Reusability describes the degree to which the application and the code in the application have been specifically designed, developed, and supported to be usable in ***other*** applications.

Score

Score As	Descriptions To Determine Degree of Influence
0	No reusable code.
1	Reusable code is used within the application.
2	Less than 10% of the application code developed is intended for use in more than one application.
3	Ten percent (10%) or more of the application code developed is intended for use in more than one application.
4	The application was specifically packaged and/or documented to ease reuse, and the application is customized at the source code level.
5	The application was specifically packaged and/or documented to ease reuse, and the application is customized for use by means of user parameter maintenance.

Hints

Hints for Rule 1

- A score of 1 is awarded for reusing code regardless of where it was developed.
- Code developed specifically for reuse within the application and used more than once within the application counts as well as code retrieved from a central library and available for general use

Hints for Rule 2

- To score 2 or more, the code must be developed for use in more then one application, stored and managed in a central library and be available for general use. Code from one application that is cut and pasted into another application is not considered reuse.
- The reusable code would be supported by documentation that enables and eases the reuse.

Hints for Rule 5

- Examples of applications customized through use of parameters include PeopleSoft and SAP and would generally receive a score of 5.
- Reused code may be ***slightly*** modified in the receiving application.
- Examples of reuse include objects or other static code maintained in an object/code library.

Typically

Scoring is not platform dependent.

11. Installation Ease

Definition

Installation Ease describes the degree to which conversion from previous environments influenced the development of the application.

Conversion and installation ease are characteristics of the application. A conversion and installation plan and/or conversion tools were provided and tested during the system test phase.

Score

Score As	Descriptions To Determine Degree of Influence
0	No special considerations were stated by the user, ***and no*** special setup is required for installation.
1	No special considerations were stated by the user, ***but*** special setup is required for installation.
2	Conversion and installation requirements were stated by the user, and conversion and installation guides were provided and tested. The impact of conversion on the project ***is not*** considered to be important.
3	Conversion and installation requirements were stated by the user, and conversion and installation guides were provided and tested. The impact of conversion on the project ***is*** considered to be important.
4	In addition to 2 above, automated conversion and installation tools were provided and tested.
5	In addition to 3 above, automated conversion and installation tools were provided and tested.

Hints

- Conversion and installation includes converting pre-existing data into new data files, loading files with actual data, or developing special installation software, such as porting.
- Purchased or developed software must be used in order to take credit for installation and conversion.

Hint for Rule 1

- Most business applications require some special setup to install the application and receive a score of 1.

Hint for Rule 2

- If the application has conversion and installation requirements and installation guides were provided, and providing these functions and guides were not on the critical path of the project, score a 2.

Hint for Rule 3

- If the application has conversion and installation requirements and installation guides were provided, and providing these functions and guides were on the critical path of the project, score a 3.

Hint for Rules 4 and 5

- If the application has conversion and installation requirements and can be installed with no external intervention, score a 4 or 5, depending on the other requirements for the scoring of 2 and 3

Typically Scoring is not platform dependent.

12. Operational Ease

Definition

Operational Ease describes the degree to which the application attends to operational aspects, such as start-up, back-up, and recovery processes.

Operational ease is a characteristic of the application. The application minimizes the need for manual activities, such as tape mounts, paper handling, and direct on-location manual intervention.

Score

Score As	Descriptions To Determine Degree of Influence
0	No special operational considerations other than the normal back-up procedures were stated by the user.
1 - 4	One, some, or all of the following items apply to the application. Select all that apply. Each item has a point value of one, except as noted otherwise. • Start-up, back-up, and recovery processes were provided, but human intervention is required. • Start-up, back-up, and recovery processes were provided, but ***no*** human intervention is required (count as two items) • The application minimizes the need for tape mounts and/or remote data access requiring human intervention. • The application minimizes the need for paper handling.
5	The application is designed for unattended operation. Unattended operation means *no human intervention* is required to operate the system other than to start up or shut down the application. Automatic error recovery is a feature of the application.

Hints

Hint to Rule 1-4a

- Application has the ability to perform start-up, back-up, and recovery; however, human response is required to initiate the function.

Hint to Rule 1-4b

- Application has the ability to perform start-up, back-up, and recovery; and no human response is required to initiate the function.

Hint to Rule 1-4c

- The application minimizes the need to access data that is not immediately available.
- This may include importing data from a distributed processor to the local processor prior to execution to eliminate access delays.

Hint to Rule 1-4d

- The application has been designed to provide the user with data in a condensed format or via a media other than paper.
- This could include elimination of detailed printed information or access to on-line reports, inquiries, microfiche, CD, or other such media.

Hint to Rule 5

- A score of 5 is assigned to an application that runs and recovers automatically from errors, on its own – an unattended operation.
- Unattended operation may include unmanned satellite, nuclear reactor, or air traffic control.

Typically Scoring is not platform dependent.

13. Multiple Sites

Definition Multiple Sites describes the degree to which the application has been developed for different hardware and software environments.

Score

Score As	Descriptions To Determine Degree of Influence
0	The needs of ***only one*** installation site were considered in the design.
1	The needs of more than one installation were considered in the design, and the application is designed to operate only under ***identical*** hardware and software environments.
2	The needs of more than one installation site were considered in the design, and the application is designed to operate only under ***similar*** hardware and/or software environments.
3	The needs of more than one installation site were considered in the design, and the application is designed to operate under ***different*** hardware and/or software environments
4	Documentation and support plan are provided and tested to support the application at multiple installation sites and the application is as described by 2.
5	Documentation and support plan are provided and tested to support the application at multiple installation sites and the application is as described by 3.

Hints The term multiple sites is a logical term and is not necessarily physical. There can be multiple sites within the same physical location. The determining factor is based upon the needs of the various installations.

Hints for Rule 0

- Most mainframe applications would probably score 0.
- However, if an application is installed on multiple mainframe computers with significantly different configurations or different operating systems, it would receive a score of greater than 0.

Hints for Rule 1

- For example, Windows NT on hardware with exactly the same configuration.

Hints for Rule 2

- For example, Windows 95, 98 and NT on hardware with a similar configuration.
- Variations could include different memory sizes, various storage capability, different processor speeds, and different printer types.

Hints for Rule 3

- For example, Windows, OS X, UNIX, Linux, and VOS3 on different types of hardware.
- Differences could include Intel based PC, MAC, Tandem, Sun, and AS400.

Typically Scoring is dependent on the number of different platforms.

14. Facilitate Change

Definition

Facilitate Change describes the degree to which the application has been developed for easy modification of processing logic or data structure.

The following characteristics can apply for the application:

A. Flexible Query

1. Flexible query and report facility is provided that can handle ***simple*** requests. (count as 1 item)
2. Flexible query and report facility is provided that can handle requests of ***average*** complexity. (count as 2 items)
3. Flexible query and report facility is provided that can handle ***complex*** requests. (count as 3 items)

B. Business Control Data

1. Business control data is kept in tables that are maintained by the user with on-line interactive processes, but changes take effect only on the ***next*** business cycle. (count as 1 item)
2. Business control data is kept in tables that are maintained by the user with on-line interactive processes, and the changes take effect ***immediately***. (count as 2 items)

Score

Score As	Descriptions To Determine Degree of Influence
0	None of the above.
1	A total of one item from above.
2	A total of two items from above.
3	A total of three items from above.
4	A total of four items from above.
5	A total of five items from above.

Hints

Flexible Query and Reporting:

- A flexible query and reporting facility means more than a list of choices in a 'canned' query or report.
- It is the ability of the user to control the data, data source, sequence and format of their query or report request.

Hints

- It means freedom to design screen layout, horizontal and vertical sorting, data item display formats, selection criteria for both files and data items.
- It includes true user programming for inquiries and is sometimes referred to as ad hoc query or reporting
- Using filters which control the amount of data viewed or printed in a fixed format is not considered as flexible query and report facility.
- Query and/or report writer capability is often provided by languages such as SQL or Focus or by some of the more dynamic ad hoc reporting tools (e.g., Crystal Reports.

Hint for Rule A1

- Simple requests may include and/or logic applied to only one internal logical file.

Hint for Rule A2

- Requests of average complexity may include and/or logic applied to **more than one** internal logical file.

Hint for Rule A3

- Complex requests may include and/or logic **combinations** on one or more internal logic files.

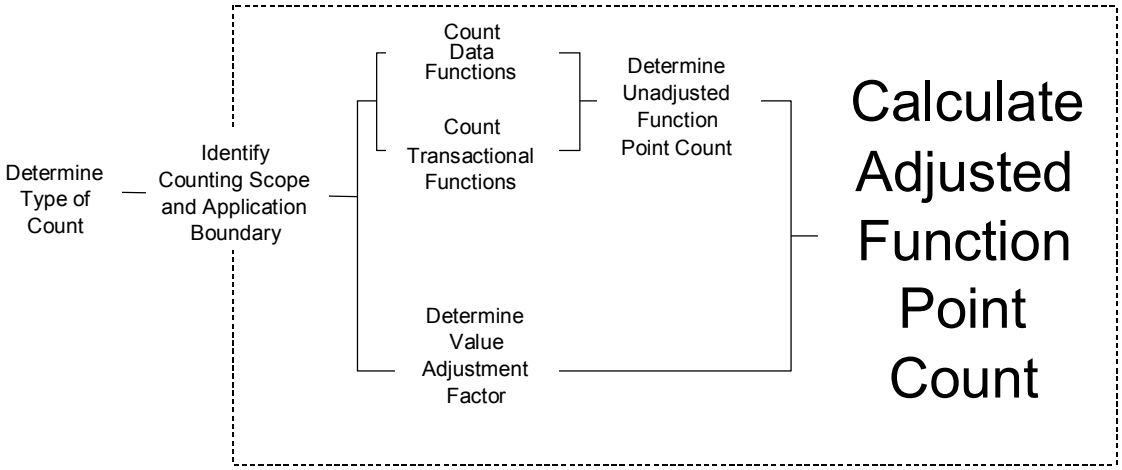
Business Control Data:

- Business Control Data (Reference Data) is stored to support the business rules for the maintenance of the Business Data; e.g., in a payroll application it would be the data stored on the government tax rates for each wage scale and the date the tax rate became effective.
- See Part 2, Code Data for additional information.

Typically

Scoring is not platform dependent.

9



Introduction

This chapter presents the formulas to complete the last step for function point analysis. It includes formulas to calculate the three types of function point counts—development project, enhancement project, and application.

Contents

This chapter includes the following sections:

Topic	See Page
Review of Steps for Function Point Analysis	**9-3**
Development Project Function Point Calculation	**9-4**
Application Functionality	9-4
Conversion Functionality	9-4
Application Value Adjustment Factor	9-4
Function Point Formula	9-5
Example: Development Project Function Point Count	**9-6**
Application Functionality	9-6
Conversion Functionality	9-8
Application Contribution to the Unadjusted Function Point Count	9-9

Continued on next page

Topic	See Page
Conversion Contribution to the Unadjusted Function Point Count	9-10
Final Calculation	9-10
Enhancement Project Function Point Calculation	**9-11**
Application Functionality	9-11
Conversion Functionality	9-11
Value Adjustment Factor	9-11
Function Point Formula	9-12
Example: Enhancement Project Count	**9-13**
Application Functionality	9-13
Application Contribution to the Unadjusted Function Point Count	9-15
Final Calculation	9-16
Application Function Point Calculation	**9-17**
Formula to Establish the Initial Count	9-17
Formula to Reflect Enhancement Projects	9-17
Example: Application Count	**9-19**
Initial Count	9-19
Count After Enhancement	9-19

Review of Steps for Function Point Analysis

The following list includes the function point analysis steps introduced in Chapter 2, Overview of Function Point Analysis.

Step	Action
1	Determine the type of function point count (Chapter 4).
2	Identify the counting boundary (Chapter 5).
3	Determine the unadjusted function point count a. Count data functions (Chapter 6). b. Count transactional functions (Chapter 7).
4	Determine the value adjustment factor (Chapter 8).
5	Calculate the adjusted function points (Chapter 9).

The remaining sections in this chapter present the formulas to complete the final step to calculate the function point count. Example calculations are included for each of the three types of function points counts:

- Development project
- Enhancement project
- Application

Development Project Function Point Calculation

The development project function point calculation consists of three components of functionality:

- Application functionality included in the user requirements for the project
- Conversion functionality included in the user requirements for the project
- Application value adjustment factor

Application Functionality

Application functionality consists of functions used after software installation to satisfy the ongoing business needs of the user.

Conversion Functionality

Conversion functionality consists of functions provided only at installation to convert data and/or provide other user-specified conversion requirements, such as special conversion reports.

For example, if a Human Resources (HR) software application was in use and a new HR application is installed, the users may require that information about employees be converted and loaded into the new application. The user-specified conversion requirement is to transfer the current employee data into the new HR system.

Application Value Adjustment Factor

The value adjustment factor is determined by using the 14 general system characteristics to rate the application functional complexity. Refer to Chapter 8 for details.

Function Point Formula

Use the following formula to calculate the development project function point count.

DFP = (UFP + CFP) * VAF

Where:

DFP is the development project function point count

UFP is the unadjusted function point count for the functions that will be available after installation

CFP is the unadjusted function points added by the conversion unadjusted function point count

VAF is the value adjustment factor

Note: After software installation, the application function point count is calculated using components of the development project function point count. See Application Function Point Calculation on page 9-17.

Example: Development Project Function Point Count

This section shows an example count for a sample development project. The project includes both application and conversion functionality.

Note: The following count is meant to be only an example and is not specifically related to other sections of this manual.

Application Functionality

The following tables show the application functionality counted for a development project.

Data Functions	RETs	DETs	Functional Complexity
Internal Logical Files			
• Job information	2	5	Low
• Suspended jobs	2	6	Low
• Report definition	1	4	Low
• Employee information	1	6	Low
External Interface Files			
• Location information	1	6	Low
• Conversion information	1	2	Low
• Window help information	1	2	Low
• Field help information	1	5	Low

Transactional Functions	FTRs	DETs	Functional Complexity
External Inputs			
Assignment report definition	1	5	Low
Add job information (screen input)	1	7	Low
Add job information (batch input)	2	6	Average
Correct suspended jobs	1	7	Low
Employee job assignment	3	7	High
EI with screen output –1	2	11	Average
EI with screen output –2	1	6	Low
External Outputs			
Jobs with employees report	4	5	Average
Employees by assignment duration report	3	7	Average
Performance review notification	3	4	Low
Weekly employees report	1	3	Low
Printed check	1	3	Low
Check transaction file	1	4	Low
External Inquiries			
List of retrieved data	1	4	Low
Bargaining Unit	1	2	Low
Field level help	1	6	Low
Weekly membership report	1	3	Low
Daily check file	1	2	Low

Conversion Functionality

The following table shows the conversion functionality for the development project.

Transactional Function	FTRs	DETs	Functional Complexity
External Input			
Employee migration	1	11	Low

Application Contribution to the Unadjusted Function Point Count

The following table shows the contribution of the application functionality to the unadjusted function point count.

Function Type	Functional Complexity		Complexity Totals		Function Type Totals
ILFs	4	Low	X 7 =	28	
	0	Average	X 10 =	0	
	0	High	X 15 =	0	
					28
EIFs	4	Low	X 5 =	20	
	0	Average	X 7 =	0	
	0	High	X 10 =	0	
					20
EIs	4	Low	X 3 =	12	
	2	Average	X 4 =	8	
	1	High	X 6 =	6	
					26
EOs	4	Low	X 4 =	16	
	2	Average	X 5 =	10	
	0	High	X 7 =	0	
					26
EQs	5	Low	X 3 =	15	
	0	Average	X 4 =	0	
	0	High	X 6 =	0	
					15
			Unadjusted Function Point Count		115

Conversion Contribution to the Unadjusted Function Point Count

The following table shows the contribution of the conversion functionality to the unadjusted function point count.

Function Type	Functional Complexity		Complexity Totals		Function Type Totals
EIs	1	Low	X 3 =	3	
	0	Average	X 4 =	0	
	0	High	X 6 =	0	
		Unadjusted Function Point Count			3

Final Calculation

Using the complexity and contribution counts for this example, the development project count is shown below. The value adjustment factor for this example is 1.05. (The formula was explained on page 9-5.)

DFP = (UFP + CFP) * VAF

DFP = (115 + 3) * 1.05

DFP = 123.9 or 124

Enhancement Project Function Point Calculation

The enhancement project function point calculation consists of three components of functionality:

- Application functionality included in the user requirements for the project
- Conversion functionality included in the user requirements for the project
- Application value adjustment factor

Application Functionality

Application functionality consists of:

- Function points identified from the functionality that is added by the enhancements
- Function points counted because existing functionality is changed during the enhancement project
- Function points counted for functionality deleted during the enhancement project

Conversion Functionality

The conversion functionality consists of function points delivered because of any conversion functionality required by the user.

Value Adjustment Factor

The two value adjustment factors are the:

- Application value adjustment factor before the enhancement project begins
- Application value adjustment factor after the enhancement project is complete

Function Point Formula

Use the following formula to calculate the enhancement project function point count.

Note: Data conversion requirements *are included* in this count.

EFP = [(ADD + CHGA + CFP) * VAFA] + (DEL* VAFB)

Where:

EFP is the enhancement project function point count.

ADD is the unadjusted function point count of those functions that were or will be added by the enhancement project.

CHGA is the unadjusted function point count of those functions that were or will be modified by the enhancement project. This number reflects the size of the functions *after* the modifications.

CFP is the function point count of those functions added by the conversion

VAFA is the value adjustment factor of the application *after* the enhancement project is complete.

DEL is the unadjusted function point count of those functions that were or will be deleted by the enhancement project.

VAFB is the value adjustment factor of the application *before* the enhancement project begins.

Note: When an enhancement project is installed, the application function point count must be updated to reflect changes in the application's functionality. See Application Function Point Calculation presented later in this chapter.

Example: Enhancement Project Count

This section shows an example for a sample enhancement project. The requirements for the enhancement project include the following changes:

- The user no longer needs to add a job online, therefore, that functionality is to be or was removed.
- The user needs to receive an additional report about jobs that includes totals.
- Additional DETs are required to add jobs in batch and correct suspended transactions. A reference to security is also added for the add job transaction.

Application Functionality

The following paragraphs explain the application functionality counted for the example enhancement project. Functionality is described as added, changed, or deleted.

Added Functionality

The following table shows the functional complexity for the added functionality counted when the project was completed.

Note: Providing a new report was an additional external output.

Transactional Functions	FTRs	DETs	Functional Complexity
External Output			
Job report	1	15	Low

Changed Functionality

The following table shows the functional complexity for the changed functionality, as the functions will exist after the enhancement project is completed.

Note: The complexity for adding a job was increased because of the additional file type referenced. The complexity for correcting suspended transactions remained low.

Transactional Functions	FTRs	DETs	Functional Complexity
External Input			
Add job information (batch input)	3	8	High
Correct suspended transaction	1	8	Low

Deleted Functionality

The following table shows the functional complexity for deleted functionality identified at the end of the project.

Transactional Functions	FTRs	DETs	Functional Complexity
External Inputs			
Add job information (screen input)	1	7	Low

Application Contribution to the Unadjusted Function Point Count

The following paragraphs explain the application functionality contribution to the total unadjusted function point count.

Added Functionality

The following table shows the contribution to the unadjusted function point count for the added functionality identified at the end of the project.

Function Type	Functional Complexity		Complexity Totals		Function Type Totals
EOs	1	Low	X 4 =	4	
	0	Average	X 5 =	0	
	0	High	X 7 =	0	
					4

Changed Functionality

The following table shows the contribution to the unadjusted function point count for the changed functionality as it will exist after the enhancement project is complete.

Function Type	Functional Complexity		Complexity Totals		Function Type Totals
EIs	1	Low	X 3 =	3	
	0	Average	X 4 =	0	
	1	High	X 6 =	6	
					9

Deleted Functionality

The following table shows the contribution to the unadjusted function point count for the deleted functionality.

Function Type	Functional Complexity			Complexity Totals	Function Type Totals
EIs	1	Low	X 3 =	3	
	0	Average	X 4 =	0	
	0	High	X 6 =	0	
					3

Final Calculation

The application value adjustment factor was 1.05 before the project began. The value adjustment factor remained the same after the project was completed.

Using the complexity and contribution counts for this example, the enhancement project function point count is shown below. (The formula was explained on page 9-11.)

EFP = [(ADD + CHGA + CFP) * VAFA] + (DEL* VAFB)

EFP = [(4 + 9 + 0) * 1.05] + (3 * 1.05)

EFP = 16.8 or 17

Application Function Point Calculation

This section provides the formulas to calculate the application function point count. There are two variations of this formula:

- Formula to establish the initial function point count for an application
- Formula to re-establish the function point count for an application after an enhancement project has changed the application functionality

Formula to Establish the Initial Count

Use the formula in this section to establish the initial function point count for an application. Initially, the user is receiving new functionality. There are no changes to the existing functionality or deletions of obsolete or unneeded functionality. The application function point count *does not* include conversion requirements.

AFP = ADD * VAF

Where:

AFP is the initial application function point count.

ADD is the unadjusted function point count of those functions that were installed by the development project.

VAF is the value adjustment factor of the application.

Formula to Reflect Enhancement Projects

When an enhancement project is installed, the existing application function point count must be updated to reflect modifications to the application. The functionality for the application can be altered in one or more ways:

- Added (new) functionality increases the size of the application
- Changed functionality increases, decreases, or has no effect on the size of the application
- Deleted functionality decreases the application size
- Changes to the value adjustment factor adds, subtracts, or has no effect on the function point count but does affect the adjusted function point count

Note: Because conversion functionality does not affect the application function point count, any conversion functionality associated with an enhancement project is omitted entirely from the application function point calculation.

Use the following formula to calculate the application function point count

after an enhancement project:

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

Where:

AFP is the application's adjusted function point count.

UFPB is the application's unadjusted function point count *before* the enhancement project begins.

Note: If this count is unavailable, it can be calculated using the formula UFBP = AFPB/VAFB; where AFPB is the adjusted application function point count before the enhancement project. VAFB is the value adjustment factor of the application before the enhancement project.

ADD is the unadjusted function point count of those functions that were added by the enhancement project.

CHGA is the unadjusted function point count of those functions that were changed by the enhancement project. This number reflects the size of the functions *after* the changes.

CHGB is the unadjusted function point count of those functions that were changed by the enhancement project. This number reflects the size of the functions *before* the changes were made.

DEL is the unadjusted function point count of those functions that were deleted by the enhancement project.

VAFA is the value adjustment factor of the application *after* the enhancement project is complete.

Example: Application Count

This section shows an example for the initial count and the count that reflects an enhancement project. Numbers for these counts are from the application count on page 9-9 and the enhancement count on page 9-13.

Initial Count

The initial application project count is shown below. The value adjustment factor is 1.05. (The formula was explained on page 9-17.)

AFP = ADD * VAF

AFP = 115 * 1.05

AFP = 120.75 or 121

Note: Only the size of the application functionality installed for the user is included in the initial count.

Count After Enhancement

The application project function point count to reflect enhancements is shown below. The value adjustment factor is 1.05. (The formula was explained on page 9-17.)

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

AFP = [(115 + 4 + 9) - (9 + 3)]* 1.05

AFP = 121.8 or 122

This page intentionally left blank.

Index to Part 1

A
Adjusted function point count
application, 9-17
development project, 9-5
enhancement project, 9-12
Application
formula, 9-17
value adjustment factor, 9-4
B
Boundary
hints, 5-6
rules, 5-5
C
Calculations
application, 9-17
conversion functionality, 9-4
value adjustment factor, 8-3
Complexity and contribution
ILF/EIF definition, 6-7
Complexity and contribution procedures
external interface files, 6-11
internal logical files, 6-11
Complexity and contribution rules
external interface files, 6-7
internal logical files, 6-7
Complexity matrices
external inputs, 7-21
external interface files, 6-11
internal logical files, 6-11
Complexity rules. See Complexity and contribution rules
Contribution rules. See Complexity and contribution rules
Control information
ILF/EIF example, 6-3
Counting hints. See Hints
Counting procedures. See Procedures
D
Data counting procedures
external interface files, 6-10
internal logical files, 6-10
Data element type. See DET
Data function types
definition, 6-1
introduction, 6-1
overview, 2-7
Definitions
complexity and contribution, 6-7
control information, 6-3
data function types, 6-1
elementary process, 6-4
external interface files, 6-3
internal logical files, 6-3
maintained, 6-4
transactional function types, 7-1
user identifiable, 6-4
Degrees of influence
end-user efficiency, 8-17
facilitate change, 8-29
heavily used configuration, 8-13
Installation ease, 8-23
multiple sites, 8-27
online data entry, 8-16
DET
definition, 6-7
DET rules
ILFs/EIFs, 6-7
Development project
conversion functionality, 9-4
Diagrams

external interface files procedures, 6-10
internal logical file procedures, 6-10
unadjusted function point count, 2-6
Difference between ILFs and EIFs, 6-3
E
EIFs. see External interface files
Elementary process
EIF example, 6-4
ILF example, 6-4
End-user efficiency, 8-17
Enhancement project
application functionality, 9-11
example count, 9-13
function point count, 4-2
value adjustment factor, 9-11
Estimated count, 4-3
Examples
control information, 6-3
elementary process for ILFs/EIFs, 6-4
enhancement project count, 9-13
ILF/EIF mandatory subgroups for RETs, 6-9
ILF/EIF optional subgroups for RETs, 6-9
maintained for ILFs/EIFs, 6-4
user identifiable for ILFs/EIFs, 6-4
External inputs
complexity matrix, 7-21
identification procedures, 7-18
translation table, 7-23
External interface files
complexity and contribution procedures, 6-11
complexity and contribution rules, 6-7
definition, 6-3
DET rules, 6-7
difference from ILFs, 6-3
example control information, 6-3
example elementary process, 6-4
example maintain, 6-4
example user identifiable, 6-4
hints to help with counting, 6-13
identification procedures, 6-10
identification rules, 6-6
mandatory subgroups for RETs, 6-9
optional subgroups for RETs, 6-9
procedure diagram, 6-10
procedures, 6-10
RET rules, 6-9
translation table, 6-11
F
Facilitate change, 8-29
File, 6-1
Final adjusted function point count
application, 9-17
development project, 9-5
enhancement project, 9-12
final calculations, 9-1
Final counts, 4-3, 9-1
Formulas
application, 9-17
conversion functionality, 9-4
establish initial count, 9-17
reflect enhancements, 9-17
Function point analysis
application calculations, 9-17
conversion functionality, 9-4
enhancement project, 4-2
formula for enhancement project application functionality, 9-11
formulas for application count, 9-17
procedures by chapter, 2-3
Function types
data, 6-1
transactional, 7-1
G
General system characteristics, 8-5
H
Heavily used configuration, 8-13
Hints
boundary, 5-6
counting EIFs, 6-13
counting ILFs, 6-13
I
Identification procedures
external interface files, 6-10
ILF/EIFs, 6-10
Identification rules
external interface files, 6-6
ILFs, 6-6
internal logical files, 6-6
ILFs. see Internal logical files
Installation ease, 8-23
Internal logical files
complexity and contribution procedures, 6-11
complexity and contribution rules, 6-7
complexity matrix, 6-11
definition, 6-3
DET rules, 6-7
difference from EIFs, 6-3
example control information, 6-3
example elementary process, 6-4
example maintain, 6-4
example user identifiable, 6-4
hints to help with counting, 6-13
identification procedures, 6-10
identification rules, 6-6
mandatory subgroups for RETs, 6-9
optional subgroups for RETs, 6-9
procedure diagram, 6-10
procedures, 6-10
RET rules, 6-9
translation table, 6-11
M
Maintained

definition, 6-4
ILF/EIF example, 6-4
Mandatory subgroups, 6-9
Manual
organization, 1-1
Matrices. See Complexity matrices
Multiple sites, 8-27
O
Online data entry, 8-16
Optional subgroups, 6-9
Organization
Counting Practices Manual, 1-1
P
Procedure diagrams
external interface files, 6-10
internal logical files, 6-10
Procedures
by chapter, 2-3
calculate value adjustment factor, 8-3
external input, 7-18
external interface files complexity and contribution, 6-11
external interface files identification, 6-10
ILF/EIF counting, 6-10
internal logical files complexity and contribution, 6-11
internal logical files identification, 6-10
steps, 2-3
R
Record element type. See RET
RET
definition, 6-9
RET rules
ILFs/EIFs, 6-9
mandatory subgroups for ILFs/EIFs, 6-9
optional subgroups for ILFs/EIFs, 6-9
Rules
boundary, 5-5
complexity and contribution for ILFs/EIFs, 6-7
DETs for ILFs/EIFs, 6-7
EIF identification, 6-6
ILF identification, 6-6
ILF/EIF, 6-5
ILF/EIF mandatory subgroups, 6-9
ILF/EIF optional subgroups, 6-9
internal logical files, 6-5
RETs for ILFs/EIFs, 6-9
S
Scope creep, 4-3
T
Transactional function types
definition, 7-1
overview, 2-8
Translation tables
external inputs, 7-23
ILFs/EIFs, 6-11
Type of count
enhancement project, 4-2
estimated and final counts, 4-3
U
Unadjusted function point count
data function types, 6-1
overview, 2-6
transactional function types, 7-1
User identifiable
definition, 6-4
ILF/EIF example, 6-4
V
Value adjustment factor, 8-1