Simple Function Point (SFP) Counting Practices Manual

Release 2.1

IFPUG
International Function Point USERS GROUP

SFP
Simple Function Point
SIMPLE AND FAST SOFTWARE SIZE
TM

This page intentionally left blank

International Function Point Users Group (IFPUG)

Simple Function Point

Release 2.1

Chairperson,
Simple Function Point Task Force
Charles Wesolowski, IFPUG Vice President
ifpug@ifpug.org

ISBN: 978-0-9903007-8-6

This page intentionally left blank

Release 2.1, October 2021

This is the first release of the IFPUG SFP published by IFPUG.

Documentation Team

SFP – Release 1

Roberto Meli, inventor

SFP – Taskforce 2020-2021

Charles Wesolowski
Daniel B. French
Esteban Sanchez
Roberto Meli
Talmon Ben-Cnaan

SFP v2.1 - reviewers

Dácil Castelo

Joe Schofield

Luigi Buglione

Tom Cagley

Sergio Brigido

Christine Green

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

This page intentionally left blank

Table of Contents

This page intentionally left blank

Preface

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.

IFPUG FPA CPM

This version of the Simple Function Point Practices Manual (SPM) was written with the purpose of being compliant and consistent with the IFPUG FPA CPM version 4.3.1 in terminology and definitions

SFP History & Background

In 2010 Roberto Meli of DPO designed a method called Simple Function Points (SiFP), that uses only two of IFPUG's Base Functional Components (BFC): the Elementary Process, and the Logical File. Initial research showed that SiFP counts can approximate IFPUG Function Points (FP) very early in the requirements development process, when the details of the domain data model (transactional and persistent data types) may not be available. An association (SiFPA) was created to publish, improve, and promote the new measurement method in many languages.

In 2019, an agreement was signed to transfer the ownership and the rights of this method to IFPUG, including the rights to improve and implement SiFP worldwide.

Release 2.1

Release 2.1 of the SiFP method is the first release developed and approved by IFPUG. The tag SiFP becomes SFP.

Future Releases

This document is meant to be a living one. We must recognize how to perform sizing in new environments as they are introduced. We need to be able to do this in the context of maintaining the validity of the sizing 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 Simple Function Points Taskforce wishes to thank all those who have helped us in our research and the production of this manual.

Charles Wesolowski

IFPUG Vice President

Chairperson, Simple Function Point Task Force

This page intentionally left blank

Introduction to the SFP Counting Practices Manual

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:

Objectives of the SFP Method

Simple Function Points is a method that produces a functional size that approximates IFPUG Function Points (FP) as described in the IFPUG Counting Practices Manual (v. 4.3.1)

This is useful very early in the software development process when details are not available to calculate the functional size of a software boundary using the IFPUG method.

These details include:

1) The Domain Data Model that gives the DET and RET counts
2) The Transactional Data Type (EI, EO, EQ) of each Elementary Process
3) The Data Function Type (ILF, EIF) of each Logical File
4) The FTR counts for each Elementary Process

This method can be used to assess the Functional Size Measure in a simplified manner that approximates the results of the IFPUG Function Point Analysis method.

SFP requires only the identification of Elementary Processes and Logical Files. It assigns a numeric value directly to these Base Functional Components (BFCs), thus significantly speeding up the functional sizing process, at the expense of ignoring the domain data model, and the primary intent of each Elementary Process.

The decision to use SFP may be influenced by factors such as the software development lifecycle phase, schedule constraints, the business use of results, available information pertaining to the software application or project, and the availability of both subject matter experts or function point analysts. It is important to realize that because SFP uses the same definitions for Elementary Process and Logicial files as the IFPUG FPA standard, it provides a sound basis for proceeding to a full function point count.

The method does not consider the Non-Functional Software Requirements and specifically the Quality Requirements and Technical Requirements. These can be measured using the IFPUG Software Non-Functional Assessment Practices (SNAP).

Throughout this manual, the term “Counting” will be used as an equivalent of “Measurement” since the measurement function consists of identifying elements to be considered and assigning a numerical value to each of them.

Characteristics of the SFP Method

The Simple Function Point method is specifically designed to be agile, fast, lightweight, easy to use, and with minimal impact on software development processes. It is easy to learn, provides reliable, repeatable, and objective results. Like IFPUG Function Points, it is independent of the technologies used and technical design principles.

Objectives of the SFP Counting Practice Manual

This document describes the Simple Function Point Sizing Method for software. The document is designed in accordance with the ISO 14143-1:2007 (R2019) standard as requested by the ISO/IEC 14143-2:2011(R2019) Information technology — Software measurement — Functional size measurement — Part 2: Conformity evaluation of software size measurement methods to ISO/IEC 14143-1.

The main goals of the manual are:

- To provide a clear and detailed description of the IFPUG Simple Function Point method.
- To facilitate the homogeneous application of the rules of the IFPUG Simple Function Point method by experienced practitioners.
- To ensure that SFP counts are consistent with the counting practices of IFPUG affiliate members.
- To guide on measuring SFP using the deliverables of popular methodologies and techniques.
- To provide a common understanding to allow tool vendors to provide automated support for IFPUG Simple Function Point counting.

Intended Audience

The method in this manual should be applied by anyone to produce a Functional Size Measure in a simplified manner that approximates the results of the IFPUG Function Point Analysis method. The manual was designed for use by persons new to Function Point Counting as well as those with intermediate and advanced experience in IFPUG Function Point Analysis as defined in the IFPUG CPM version 4.3.1.

Organization of the SFP Counting Practices Manual

There are three major parts in the SFP Counting Practices Manual (SPM)

- Part 1: The SFP Method
- Part 2: Examples
- Part 3 Appendices and Glossary

Part 1 – The SFP Method	This section contains a description of the IFPUG SFP method. It includes the principles on which it is based • scope of measurement; • the software measurement model; • description of the types of Base Functional Components (BFC); • operating procedure for measurement; • formulas for the aggregation of the elementary values; • documentation standards for the measurement.
Part 2 – Examples	This section contains examples of the use of the method.
Part 3 – Appendices and Glossary of Terms	This section contains additional information and the reference glossary for the software measurement activity.

Manual Revision Process

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

Frequency of Changes

During January of each year, a new version of the SFP Counting Practices Manual may become effective. It will include any new or changed definitions, rules, or practices that have been finalized by the Functional Sizing Standards Committee (FSSC) since the previous version. These updates correspond to major releases.

From time to time, it may be necessary to update the Manual, either for editorial reasons (correction of grammatical errors, style variations, usability improvements, reorganization of content, etc.) or for improvements, such as adding examples, changing descriptions, and adding/updating diagrams. These updates correspond to minor releases.

Change Process

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

Step	Action
1	The issue is submitted to the FSSC.
2	The issue is assigned for research.
3	The FSSC reviews and discusses the issue.
4	The FSSC 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 Manual.

1
Issue Submitted

The reader submits ideas, changes, or issues to the Functional Sizing Standards Committee by sending an email to ifpug@ifpug.org

2
Research Assigned

A member of the FSSC is assigned the responsibility for identifying all alternatives, the rationale, and the potential impact of each alternative if it is implemented. A 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.

3 FSSC Review	The FSSC 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.
4 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.
5 Impact Study Initiated	The FSSC 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 FSSC receives written feedback indicating there is little or no impact, the study is discontinued.
6 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.
7 Decision Communicated	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.
8 Decision Effective Date	The SFP Counting Practices Manual will be updated to reflect the decision with the next release.

Training Requirements

Usability evaluations of this publication have verified that reading the SFP Counting Practices Manual alone may not be sufficient to apply the process at the optimum level. Training is recommended, particularly for those new to Function Point Analysis.

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 SFP 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

Used Documentation

The following documentation was used to develop the present release:

- SFP-01.00-RM-EN-01.01 - Simple Function Point Functional Size Measurement Method Reference Manual, SFPA 2014
- International Function Point Users Group (IFPUG) Function Point Counting Practices Manual" (CPM), Release 4.3.1, this IFPUG manual has been used to re-use definitions and other items to align IFPUG SFP with the current IFPUG FP method.
- ISO 14143-1:2007 (R2019) "Information technology – Software measurement – Functional size measurement – Part 1: Definition of concepts" and ISO/IEC/IEEE 24765:2010 "Systems and software engineering – Vocabulary", as standard references for Functional and Non-Functional Requirements definitions.
- ISO/IEC 14143-2:2011 (R2019) "Information technology — Software measurement — Functional size measurement — Part 2: Conformity evaluation of software size measurement methods to ISO/IEC 14143-1"
- ISO/IEC TR 14143-5:2004 (R2019), Information technology -- Software measurement -- Functional size measurement -- Part 5: Determination of functional domains for use with functional size measurement

Other Related IFPUG Documentation

The following table describes other IFPUG related publications.

Document	**Description**
IFPUG Glossary (Available with CPM, SPM 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.

This page intentionally left blank

Part 1 –The SFP Method

This page intentionally left blank

Part 1 Chapter 1

Description of the Method

Introduction This chapter presents a description of the objectives and benefits of SFP and the Measurement Framework for Simple Function Point Analysis.

Contents This chapter includes the following sections:

General Concepts

Per the prescriptions of the ISO/IEC 14143-1:2007 (R2019) international standard, user requirements, related to a software application, can be grouped into three main classes: Functional Requirements, Technical Requirements and Quality Requirements. The second and third are also known as Non-Functional Requirements (NFR). Functional measurements of software (FSM – Functional Size Measurements), to which Simple Function Points belongs, are related exclusively to the first of the three categories.

The purpose of the Simple Function Point method is to provide an objective measurement of the number of functions that a software application offers to its users (humans and/or other software systems) by quantifying "what" it makes possible, in terms of available data and operations upon them.

The principal component of a functional software measurement method is the concept of a Base Functional Component (BFC). The ISO/IEC 14143-1:2007 (R2019) standard defines the BFC as an "elementary unit of Functional User Requirements...". The term "elementary" is, in this context, a synonym of "atomic" — in the original sense of the philosophical term - meaning that "it cannot be further decomposed". In the Simple Function Point method, subcomponents (such as the data attributes that make up a logical file or a transaction) are not identified. The BFC is the elementary entity to which numeric values are attributed, based on the measurement function.

There are three basic categories of functional requirements: the requirements that represent data flows or movements, those that represent data processing rules, and those relating to permanent or persistent data storage. The IFPUG SFP method identifies and gives a value to only the first and the third category of user requirements: logical transactions that move data (logical flows) and the logical files that keep them for an undefined period. The specific nature of logical processes (algorithms – calculations - transformations, etc.) is not expressly and separately measured.

The IFPUG SFP method adopts the assumption that the functional value of a software product or object[1] is proportional only to the number of logical transactions and to the number of logical files required. No identification of subcomponents is needed to calculate the SFP value of each BFC.

Organizations can apply this method to measure the size of a software product to:

- support quality and productivity analysis;
- estimate cost and resources required for software development, enhancement and maintenance;
- provide a normalization factor for software comparison;
- determine the size of a purchased application package by functionally sizing all the functions included in the package;
- assist users in determining the benefit of an application package to their organization by functionally sizing functions that specifically match their requirements.

[1] In this document, the term software "object" is used generically to mean a computer program accompanied by documentation that describes it. There is no specific reference to Object-Oriented Programming

Simple Function Point analysis measures software by quantifying the tasks and services (i.e., functionalities) that the software provides to the user based only on logical design. The objectives of Simple Function Point analysis are to measure:

- functionality implemented in software, that the user requests and receives;
- functionality impacted by software development, functional enhancement independently of the technology used for implementation.

The process of Simple Function Point analysis is:

- simple enough to minimize the overhead of the measurement process;
- consistent among various projects and organizations.

In order to effectively apply this method, persons can be formally trained in the method using IFPUG Certified course materials.

This document is one component of the IFPUG publications. It is recommended that this document be read in conjunction with the other IFPUG publications. These publications guide the application of the rules described within this method and background information to aid in understanding the use and applicability of the resulting functional size.

IFPUG website at www.ifpug.org.

Application Domains

The IFPUG SFP method applies to a wide range of application domains as defined in the ISO/IEC TR 14143-5:2004 Technical Report. Specifically, it applies to an application that may be classified as data-rich, algorithm-rich, and control/communication-rich. As non-exhaustive examples: Management Information Systems (MIS), Decision Support Systems DSS, Data Warehouse Systems, Geographical Information Systems (GIS), Web applications, Numerical Control Systems, Internet of Things (IoT), Mobile applications, and Embedded systems.

The Software Model On Which The Measurement Is Based

SFP is a method designed to measure a software application. A software application may be described by many artifacts (business requirements, functional specifications, technical design, etc.). The software artifacts on which the measurement is operated must be identified to adequately apply the sizing function. A software artifact is the result of a process organized around cycles, phases, sprints, activities, tasks, etc. At the end of each of these steps, the software artifacts should be in a consistent state. The functional measurement of a software application may be taken whenever adequate information about functional requirements is available: to be considered adequate, the information must be complete, unambiguous, and at the level of granularity corresponding to the SFP BFC definitions. During the production process, the IFPUG SFP measures should change as the Functional User Requirements change. If the functional requirements remain the same during the production process, also the SFP measure remains the same.

Size measurement is a measurement of the software application seen as a product, not of the process that has produced it. It is a software product-oriented measure that can be derived at any stage of the software development lifecycle, given the appropriate information.

There are three types of possible measurements:

1. **Development project simple function point count**: the size of software involved by the development project to create a new software application;
2. **Enhancement project simple function point count**: the size of software involved by the enhancement project to "functionally enhance" an already existent application;
3. **Application/Baseline simple function point count**: the size of an application as an asset (after the development and after any following functional enhancement).

In order to avoid misunderstandings on the terms used in this manual, it is necessary to clarify the concepts of "maintenance" and "enhancement".

By "maintenance" we intend the "process of retaining a hardware system or component in, or restoring it to, a state in which it can perform its required functions" (ISO/IEC/IEEE 24765:2017 Systems and software engineering-Vocabulary).

According to the ISO/IEC 14764:2006, maintenance can be classified into 4 main classes:

- Corrective
- Preventive
- Adaptive
- Perfective

By "maintenance enhancement" we intend the "modification to an existing software product to satisfy a new requirement (IEEE 14764-2006 Software Engineering - Software Life Cycle Processes - Maintenance, 3.5) Note: There are two types of software enhancements, adaptive and perfective. A maintenance enhancement is not a software correction."

A maintenance enhancement may involve a functional enhancement and/or a non-functional enhancement. The Simple Function Point method may be used to measure any change in functionality for any type of maintenance in which it takes origin.

- **Functional Enhancement** is so defined as a set of activities aimed at creating new features/functionalities to be added to an existing software application, and/or modifying /deleting existing ones;
- **Non-functional Enhancement** is defined as a set of activities aimed at improving non-functional performance (usability, maintainability, etc.) or to respect technical or quality requirements with no changes in the functionalities.

In the context of this manual and to maintain compliance with the IFPUG Counting Practices Manual, when we use the "Enhancement" term we refer exclusively to **Functional Enhancement.**

Simple Function Point applies to a newly developed software application or to a software application changed by functional enhancement or to a software application considered as an asset[2]. For other types of maintenance that do not imply a change in the functional requirements, it is not possible to "calculate" a functional measure. These can be measured using IFPUG SNAP.

Simple Function Point is a product size measurement method, not a process measurement method, therefore:

- the "measurement of the development project" is meant as a "measurement of the software functionalities provided by the development project"
- the "measurement of the enhancement project" is meant as “measurement of the software functionalities ‘worked’ (added, changed or deleted) by the enhancement project”
- the "measurement of application/baseline" is meant as the "measurement of the software functionalities that are available after the initial development and the eventual subsequent enhancement activities". It is an "asset" type value.

2 "Software asset" refers to the set of software developed and available to an organization for use in an enterprise or institutional information system.

A Generic Software Model

The following illustration represents the functional requirements for "moving" data, "processing" data, and "storing" data. The gray circle represents the software application and its boundary.

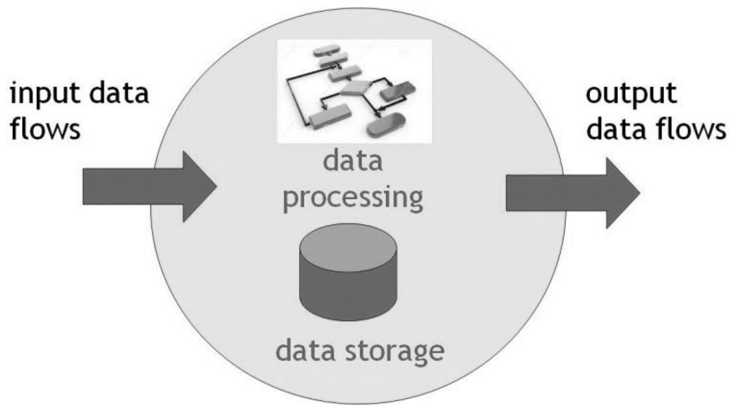


Figure 1

An **application** is a cohesive collection of automated procedures and data supporting a business objective; it consists of one or more components, modules, or subsystems. The **boundary** is a conceptual interface between the software under study and its users.

A **user** is any person or thing that communicates or interacts with the software at any time. A software application may be considered as a user by another application if they exchange information at the same level of logical representation, like in a peer-to-peer exchange. Incorporation of an internal or external component or service into the business functionality required by the user is not considered as a case of “communication” between software applications. For example, a bank deposit application may exchange money movements data flows with a surveillance application of a central bank organization. The bank deposit application is considered as a user of the surveillance central bank application and vice versa. An e-commerce application that uses a bank service to manage a payment to terminate the acquisition transaction is an incorporated functionality and we do not consider the API as a user of the e-commerce application. A **user view** is the Functional User Requirements as perceived by the user. **Functional User Requirements** are a sub-set of the user requirements specifying what the software shall do in terms of tasks and services.

The definition/identification of the application boundary is a preliminary mandatory step for any measurement to identify identical BFCs that will be measured only once per application even if encountered again in the functional menu structure for the use of a software system. The same BFCs present in several applications must be measured for each application that contains them, instead.

An Application Catalog or List documents the boundaries among the applications of the global informational asset that will form, therefore, the basis for all the measurements that will be done on this asset. The eventual redefinition of the boundaries among applications, desired for any organizational or business reason, usually changes the values of the functional measurements of the total asset for the same features provided to users, because of duplication of elements among the various applications. For example, if you merge two applications that share a set of logical files, the data elements that were counted twice before now count only for one due to the rule of identification of identical BFCs. Since this activity, therefore, makes it impossible to compare the values of assets calculated before and after the change of boundaries, it should be carried out only in exceptional cases. For example, when the organization wants to sell part of an asset to a different company, or when an application is likely to become huge due to large functional enhancements not originally expected, and so on. A redefinition of boundaries should be done with the awareness that the overall asset evaluation (and the related indicators, SLA, and dashboards) will be probably different (lower or higher) without any changes in the software (functional requirements). The boundaries among applications should remain stable to provide continuity and congruence to the measurements of the global asset.

In common IT practices, there are various ways to combine elementary software elements into software applications. Many of these derive from technical viewpoints and are not adequate for Function Point Analysis. For example, components that share allocation on a technology platform or layer may be combined to identify an application in IT practices. A functional method of measurement should use criteria related to a "logical" or user point of view of the services provided by a software application and not related to technological architecture. For this reason, it is important that an organization defines its own catalog of applications from the functional measurement perspective. An essential task in governing software measurement is, then, to maintain a mapping between the catalog of (measurable) software applications and the various operational and technical catalogs that eventually already exist.

BFC Types

The IFPUG SFP method uses only two BFCs:

- LF: Logical File
- EP: Elementary Process

1. LF: Logical File

"A Logical File represents functionality provided to the user to meet internal and external data storage requirements. It is a user recognizable group of logically related data or control information maintained and/or referred within the boundary of the application being measured.”

The term file here does not mean physical file or table. In this case, file refers to a logically related group of data and not the physical implementation of those groups of data. **Control Information** is data that influences an elementary process by specifying 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.

The term **User Recognizable** refers to requirements for processes and/or 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 have the functionality to maintain and store Employee information in the application.

When identifying the LFs it must be clear that two different types of sets of data can be found in the user requirements, which are called:

- Functional data group
- Non-functional data group

Functional data groups are used to satisfy the Functional User Requirements. For example, the following can be identified as functional data sets: Clerk, Sale, Supply contract, Car, Blast furnace, Missile, Bank Account.

Non-functional data groups are sets of data aimed at implementing Non-Functional Requirements such as usability (data for drop-down lists, numeric range limitations, set of constant admitted values, code tables, stylesheets, etc.) or performance (data access indexes) or maintainability (configuration parameters), and so on.

Only the first category (the functional data group) can therefore be identified as an LF.

2. EP: Elementary Process

"***An Elementary Process is the smallest unit of activity which is meaningful to the user, that constitutes a complete transaction, it is self-contained and leaves the business of the application being measured in a consistent state***".

The term ***transaction*** here does not mean a physical collection of software instructions grouped according to a technical criterium (a Non Functional Requirement). An elementary process is, instead, a logical aggregation of processing steps which is **meaningful** from a logical user perspective and it is fulfilling a Functional Requirement.

The property of **completeness** is associated to the achievement of a user unitary goal. An EP includes all the mandatory and optional processing steps that are considered indispensable to achieve that goal. They include the requirements specifically requested by the user to complete an elementary process such as validations, algorithms or calculations and reading or maintaining a data function. They also include any application response message

Self-contained means that no prior or subsequent processing steps are needed to initiate or complete the Functional User Requirement associated to the EP.

The key concept in the definition is the **consistency** of the application "state". An application is in a consistent state when all the constituting parts (especially the contents of the Logical Files) are coherent and not in mutual contradiction. A consistent state is a point at which an EP has been fully executed; the related Functional User Requirement has been satisfied and there is nothing more to be done.

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.

Every time an enhancement project requires changes to some portion of the Processing logic, or the Logical File, this is not sufficient reason to decompose the EP impacted into multiple different EP´s if the business process objective to be achieved by that EP remains the same.

Part 1 Chapter 2

The Counting Process

This chapter presents a summary of the SFP counting process and its steps.

Contents

This chapter includes the following:

Introduction

The following illustration is a diagram of the measurement process that is explained in detail later.

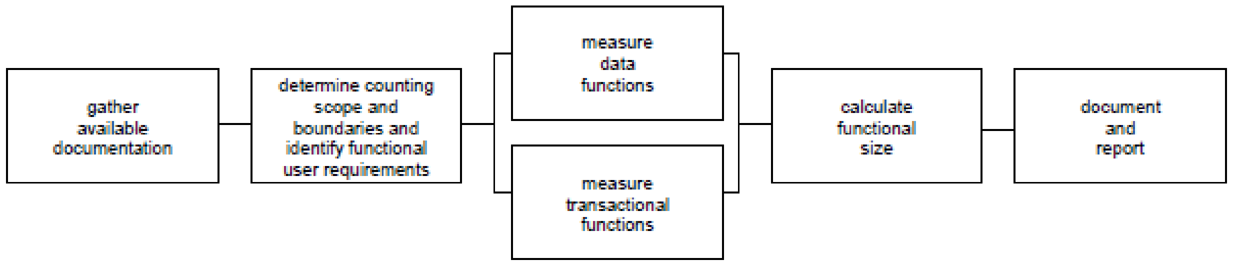

gather available documentation
determine counting scope and boundaries and identify functional user requirements
measure data functions
measure transactional functions
calculate functional size
document and report

Figure 2 - Measurement process

Gather Available Documentation

This step involves gathering all the information needed for a reliable functional measurement. The IFPUG SFP method is not dependent on any technology or analysis, design, and user requirements' representation method. The measurement analyst, before performing the actual measurement, must put in place a searching process to locate all design or operational documents and the people who can be useful in the subsequent steps of the method. For as much as IFPUG SFP measurement is independent of the way user requirements are represented, it is still true that having the "right" documents and people for the measurement needs can facilitate or hinder the measurement productivity and quality. The measurement requires completeness and granularity of the functional requirements. Functional requirements must achieve a level of refinement that allows the identification of each BFC with certainty, based on the logical uniqueness of each BFC.

Similarly, information should be available to establish the boundaries of the applications.

Useful Project/Application Information Sources

In general, the following sources are useful when conducting any functional size measurement:

• Requirements document(s)

• Entity relationship diagrams

• Object models

• Data models

• File and database layouts (with logical, user required attributes identified)

• Interface Agreements with descriptions of batch feeds/transaction files

and interfaces to/from other applications

• Samples of reports, on-line screens, and other user interfaces

• Demonstration of application operation

• One or more application experts (for the application being measured)

• One or more customers/application users (on-call during the counting

session)

• User guide, training manuals, and application help

• System design documentation

• Functional specification

• Use cases

Note: The list above is not all-inclusive.

Determine the Counting Scope and Boundary and Identify Functional User Requirements

Identify the Boundaries of the Applications Involved in the Measurement

The boundary is a conceptual interface between the software under study and its users.

The boundary (also referred to as application boundary):

- Defines what is external to the application
- Indicates the border between the software being measured and the user
- Acts as a ‘membrane’ through which data processed by transactions (EPs) pass into and out from the application
- Encloses the logical data maintained by the application (LFs)
- Assists in identifying the logical data referenced by but not maintained within this application (LFs)
- Is dependent on the user’s external business view of the application.
- It is independent of technical and/or implementation considerations

The positioning of the boundary between the software under investigation and other software applications may be subjective. It is often difficult to delineate where one application stops and another begins.

As stated before, the identification of the applications and their boundaries is driven by logical principles, not technical, and focused on the user's standpoint (at any level of abstraction it is). The focus is on what the user can understand and describe.

The following rules must be applied:

- 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.

Determine the measurement purpose, scope, and type

A specific measurement exercise is characterized by a "scope", which may relate to one or more applications. The choice of measurement scope does not automatically redefine the boundaries among the applications.

The scope of measurement defines the user features that will be included in a specific IFPUG Simple Function Points measurement. The scope:

- defines a (sub)set of the software to be measured;
- is determined by the goal of for the measurement activity;
- identifies which user functions should be included in the measurement for providing the required answers of the measurement goal;
- it may include more than one software application. If so, multiple application boundaries would be identified.

For example, a certain project may relate, at the same time, to the development of a new application and functional enhancement of pre-existing applications that must interface with it. The scope of the measurement includes both the software made from scratch and the pre-existing software that has been changed. The measurement must be made for each application separately and then the IFPUG SFP values might be summed up to have an answer to the business expectation.

The scope is closely related to the measurement purpose in the sense that it is determined by it. The purpose of the measurement is, generally, a knowledge goal aimed at a management level and does not affect the measurement rules but only how the measurements are split, joined, transformed, and connected to each other. For example, if a knowledge goal was to compare the level of online interactive features to the batch processes for each application of the catalog, the scope would exclude the measurement resulting from the data part and would separate the online BFC measurements from the batch BFC measurements.

The purpose of the measurement determines also the type of measurement: development, enhancement, application/baseline.

In order to move to the next steps, it is necessary to identify all the Functional User Requirements required for the software under measurement.

Measure Data Functions

Identify the LFs from the gathered documentation, filling a LFs directory. Logical sets of data that are used in any way by the EPs of the application should be treated as LFs. There is no difference between the LFs that are only read or read and written by the EPs.

A Logical File function represents functionality provided to the user to meet internal and external data storage requirements.

Note: Data functions are most easily identified using a logical data model; however, this does not preclude the use of the measurement process in environments where alternative data or object modeling techniques are employed. Data modeling terminology is used to document the data function rules, but the same approach can be applied with other techniques.

To identify data functions, the following activities shall be performed:

- ❑ Identify all logically related and user recognizable data or control information within the counting scope
- ❑ Exclude entities that are not maintained by any application
- ❑ Group related entities that are entity dependent

  Note: Entities that are entity independent are considered to be separate logical groups of data.
- ❑ Exclude those entities referred to as Non-functional data groups
- ❑ Exclude entities that do not contain attributes required by the user

- ❑ Remove associative entities that contain additional attributes not required by the user and associative entities that contain only foreign keys; group foreign key attributes with the primary entities

  Note: Foreign key attributes are data required by the user to establish a relationship with another data function.

Uniqueness Rule

Each LF must appear once and only once in the application organized directory. Within an application, two LFs are identical when they refer to the same object of interest to the user.

An LF must be counted in each application in which it is used.

Measure Transactional Functions

Using the gathered documentation, identify the EPs filling an EPs directory.

The Elementary Process Identification rules incorporate concepts that represent criteria to compose or decompose business process activities, reaching the smallest unit of activity level that:

- ❑ is meaningful to the user

  For example, the functional user requirement requires the ability to add a new employee to the application.

- ❑ constitutes a complete transaction

  For example, the user definition of employee includes salary and dependent information. If the number of dependents is greater than zero, adding an employee must include dependent information. In this example, adding an employee (without adding address, salary and dependent information) does not meet all of the criteria. Other systems may treat maintenance of salary and/or dependent information independently from employee.

- ❑ is self-contained

  For example, the add process is not self-contained unless all mandatory information is entered and all the processing steps are included; e.g., validations, calculations, updating the ILFs.

- ❑ leaves 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.

Identify an elementary process for each unit of activity identified that meets all of the above criteria.

Uniqueness Rule

It´s not enough to identify an EP, but also to ensure, only after the EP has been identified, that it isn´t duplicated when compared with others EP already identified, following the same criteria applied before. The uniqueness in comparison to other Elementary Processes represents another important concept associated with an Elementary Process that must be understood in the context of business processes. Each EP must appear once and only once in the organized directory of BFCs for the application. Within one application two EPs are identical when they perform the same processing on the same data and could be used interchangeably. The functional design normally identifies identical EP candidates.

An EP must be counted in each application in which it is used.

Calculate Functional Size

Once the EP and LF directories are complete, the scores are assigned to the individual BFCs and added together as shown below. The scores to be assigned to each BFC are:

LF = 7.0 SFP

EP = 4.6 SFP

Document and Report

The measurement must be documented with all the assumptions and measurement decisions taken, the standards used, the guidelines adopted, the links to the design documentation as set forth in the special section of this Manual (Part 1 Chapter 4).

This page intentionally left blank

Part 1 Chapter 3

Formulas for Calculation of the Functional Measure

Introduction In IFPUG SFP method there are different formulas to calculate the functional measure, depending on the type of measurement required.

Contents This chapter includes the following:

Development

When creating a new application, two components will be considered for the software involved by the activity: new (ADD) and *conversion* (CFP) features supporting start-up of usage of the application, such as logical files population. The latter will be part of the measurement of the features released by the development activity but not of the baseline measurement after development.

DSFP = ADD + CFP

Application or Baseline after the initial development

At the end of the development activity, the measurement of the baseline of the application released (ASFP) will be that of the activity that has generated it (DSFP) minus the conversion (CFP) component.

ASFP = ADD

Enhancement

For an Enhancement activity of an existing application, there are 4 components to be considered for measurement: features that were added (ADD), those changed (CHG), those deleted (DEL), and the conversion functions (CFP).

ESFP = ADD+CHG+DEL+CFP

Application or Baseline after an enhancement

After any Enhancement activity, the measurement of the baseline of the application released (ASFPA) will be that of the baseline measurement before the project activity that created it (ASFPB) plus the new features (ADD) and minus the features removed (DEL).

ASFPA = ASFPB + ADD – DEL

Part 1 Chapter 4

Documentation of the Functional Measurement

Introduction

The last step of the IFPUG SFP functional measurement procedure is to document and present the measures. Below is the minimum set of information needed for this task.

Since software measurement is driven by a business goal or question that determines the scope of measurement, which may involve more than one application, the measurement document must have a modular structure that includes a common part and a part that is repeated for each application involved in the scope.

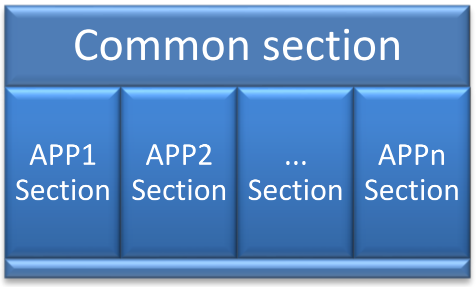


Figure 3 - Documentation

Contents

This chapter includes the following:

Common Section

- Executive summary
- Objectives of the measurement
- Users of the system
- Scope of the measurement
- Date of the report
- Report authors
- Personnel involved in measurement activity
- Versions of the method
- Documentation used to develop the measure

For each Application

- Executive summary
- Application Identification
- Type of specific measurement
- Release date of the measurement
- Date of approval of the measurement
- Authors of the measurement
- Personnel involved in measurement and their role and position
- Specific document references on which the measurement was based
- List of BFCs with corresponding functional weights
    - LF section (with link to the related functional requirement)
    - EP section (with link to the related functional requirement)
- Results (calculation formula)
- List of constraints, assumptions, and critical aspects
    - overall
    - for each BFC

Part 1 Chapter 5

Convertibility with IFPUG FP

Introduction Convertibility has been studied for version 4.x of the IFPUG FP method (meaning from version 4.0 to version 4.3.1). The results are described below.

Contents This chapter includes the following:

Methodological Correspondence

Analysis of the theoretical correspondence between elements of the IFPUG FP method and the IFPUG SFP method found the following evidences.

Correspondence of Concepts and Types of Measurement

The application, scope, boundary, and purpose concepts of the two measurement methods are identical. The types of measurement (development, enhancement, baseline) are identical. This leads to the identification of the same “entities” to be measured.

BFC Correspondence

The definition of an elementary process for the IFPUG FP and the IFPUG SFP method is the same, so every IFPUG FP transactional type element (EI, EO, EQ) corresponds to a SFP EP. Every IFPUG FP data type element (ILF, EIF) corresponds to a SFP LF.

From these considerations, it appears that the list of BFCs that would be produced by the identification phase in the IFPUG FP method has the same numerosity and general type correspondence (transactional and data) as that of IFPUG SFP.

Correspondence of Calculation Formulas

The calculation formulas for development activity, for assets after development, and functional enhancement maintenance are similar. There is a difference only in the formula used to update baselines after functional enhancement since there is no change in complexity.

Calculation of SFP measures starting from FP measures

In order to derive a SFP value from an FP value the following formula may be used:

SFP = (#EI + #EQ + #EO) x 4.6 + (#ILF+ #EIF) x 7

Empirical Correspondence

It is not possible to algorithmically derive an FP measure from a SFP measure. Any SFP EP may be an EI, EO, EQ and any SFP LF may be an ILF or EIF. So, given a list of EPs, it is not possible to derive a corresponding list of EIs, EOs, EQs with their complexity and the same is true for LFs and ILFs, EIFs. It is necessary to go back to the functional requirements in order to look for the detailed information needed to complete an FP calculation.

Conversion between a SFP measure and an FP measure of the same Functional User Requirements may only be based on statistical models based on empirical evidence. The analysis of the correlation between SFP measures and FP measures is out of the scope of this manual and it is a live research subject.

Part 2 – Examples

This page intentionally left blank

Part 2 Chapter 1

Examples

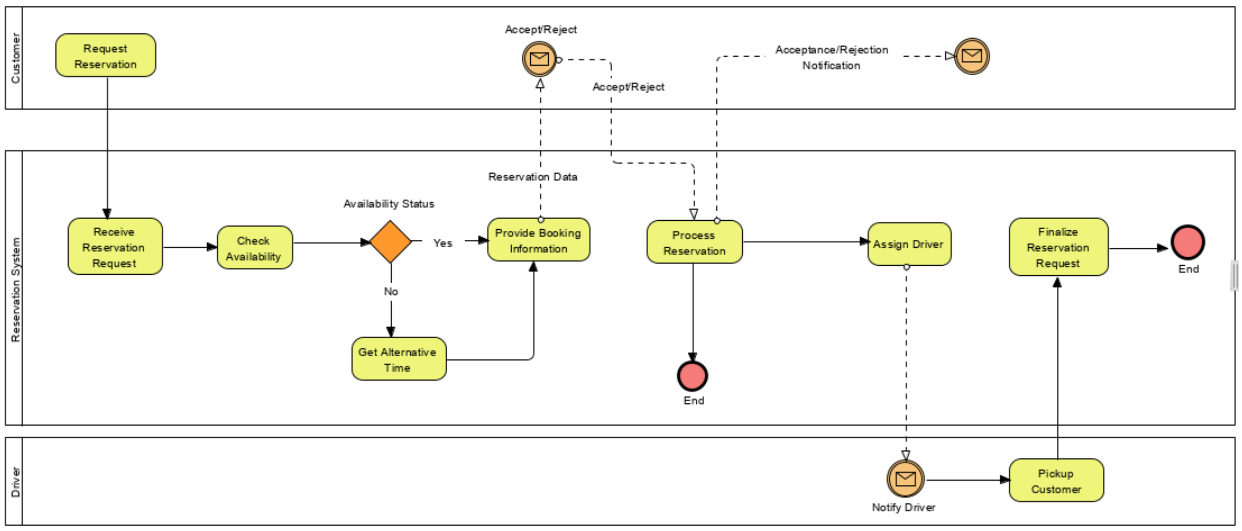
Introduction This chapter includes several examples of how Functional Requirements can be measured using the SFP method.

Contents This chapter includes the following examples:

Example 1: EPs identification

The following example illustrates the EP identification step. The diagram below represents the business process of a “**Reservation System**”* at an intermediate level of detail, whose business objective is to enable ***Customers to reserve a car ride.***

The example was extracted from the IFPUG White Paper "Function Point Analysis Rules Interpretations for Elementary Processes", which was in the approval process on the date of the SFP Manual first publication. Changes may be expected in future releases of the manual depending on how the example will be constituted when the White Paper is published.



** The process diagram is just an example and doesn´t meant to represent similarities with any real car reservation processes. Database, Hardware or software specifics for the Reservation System are not in consideration for the purpose of the example.*

The process activities are described at the table below:

Activity Name	Activity Description
Request Reservation	A Customer sends a reservation request to the Reservation System providing reservation data attributes: date, time, pickup address and destination address.
Receive Reservation Request	The Reservation System receives Customer reservation request and records request data in the Reservation database
Check Availability	The Reservation System searches for request data availability in the Reservation database. The Reservation System sets Availability Status to “Yes” if the search finds an available date/time slot that matches the request or sets Availability Status to “No” if the date/time slot is not available.
Get Alternative Time	If Availability Status = “No”, the Reservation System searches the Reservation database for the closest available date/time slot for the requested pickup and destination addresses.
Provide Booking Information	The Reservation System sends an automatic notification to the Customer providing reservation data based on original request (if Availability Status = “Yes”) or providing the closest available date/ time slot available (if Availability Status = “No”).
Process Reservation	The Customer replies to the Reservation System, Accepting or Rejecting the reservation data. The Reservation System records the response in the Reservation database. If the response notification is “Accepted”, the Reservation System notifies the Customer about the reservation data details. If the response notification is “Rejected”, the Reservation System acknowledges the rejection to the Customer and the process ends.
Assign Driver	If the customer response notification is “Accepted”, the Reservation System assigns a driver to the confirmed date, time, pickup and destination addresses reserved. The Reservation System records assignment in the Reservation database and notifies the assigned driver.
Pickup Customer	The Driver picks up the Customer on date, time and pickup address reserved and sends pickup confirmation to the Reservation System.
Finalize Reservation Request	The Reservation System marks the reservation as completed in Reservation database and the process ends.

The actors “Customer” and “Driver” are the Users of the Reservation System. Therefore, the Reservation System business value is represented by their User View of the reservation process, which is the one of the bases for Elementary Process identification.

The table below provides the analysis of each business process activity in order to identify the Elementary Processes, by applying the EP identification rules as stated above:

	Elementary Process Identification Rules			
Business Process Activities	**Meaningful to the user?**	**Constitutes a complete transaction?**	**Self-contained?**	**Leaves the business of the application being counted in a consistent state?**
Request Reservation	Yes. Request Reservation is part of the Functional User Requirements.	No. Request Reservation alone isn´t a complete transaction because the process must also receive request information, check availability, get alternative date/time and provide reservation information to the Customer. These steps cannot be logically separated.	No. Receive request, Check Availability, Get Alternative Date/Time and Provide Reservation information are all subsequent processing steps necessary to complete the elementary process.	No. The complete business need is only achieved when the Request Reservation information is submitted, received, processed and returned to the customer.
Receive Reservation Request	Yes. Receive Reservation Request is part of the Functional User Requirements.	No. Receive Reservation Request alone isn´t a complete transaction because the process must also request reservation, check availability, get alternative date/time and provide reservation information to the Customer. These steps cannot be logically separated.	No. Request Reservation is a prior processing step that must be performed. Check Availability, Get Alternative Date/Time and Provide Reservation information are all subsequent processing steps necessary to complete the elementary process.	No. The complete business need is only achieved when the Request Reservation information is submitted, received, processed and returned to the customer.
Check Availability	Yes. Check Availability is part of Functional User Requirements.	No. Check Availability alone isn´t a complete transaction because the process must also submit reservation request, receive request information, check availability, get	No. Request Reservation and Receive Request are both prior processing steps that must be performed. Get Alternative Date/Time and	No. The complete business need is only achieved when the Request Reservation information is submitted, received, processed and

	Elementary Process Identification Rules			
Business Process Activities	**Meaningful to the user?**	**Constitutes a complete transaction?**	**Self-contained?**	**Leaves the business of the application being counted in a consistent state?**
		alternative date/time and provide reservation information to the Customer. These steps cannot be logically separated.	Provide Reservation information are all subsequent processing steps necessary to complete the elementary process.	returned to the customer.
Get Alternative Time	Yes. Get Alternative Time is part of the Functional User Requirements.	No. Get Alternative Time alone isn´t a complete transaction because the process must also submit reservation request, receive request information, check availability, and provide reservation information to the Customer. These steps cannot be logically separated.	No. Request Reservation, Receive Request, and Check Availability are all prior processing steps that must be performed. Provide Reservation information is a subsequent processing step necessary to complete the elementary process.	No. The complete business need is only achieved when the Request Reservation information is submitted, received, processed and returned to the customer.
Provide Reservation Information	Yes. Provide Reservation Information is part of the Functional User Requirements.	No. Provide Reservation Information alone isn´t a complete transaction because the process must also submit reservation request, receive request information, check availability, and get alternative date/time. These steps cannot be logically separated.	No. Request Reservation, Receive Request, and Check Availability, and Get Alternative Date/Time are all prior processing steps that must be performed in order to complete the elementary process.	No. The complete business need is only achieved when the Request Reservation information is submitted, received, processed and returned to the customer.
Process Reservation	Yes. Process Reservation is	Yes. Process Reservation	Yes. Process Reservation	Yes. The complete

	Elementary Process Identification Rules			
Business Process Activities	**Meaningful to the user?**	**Constitutes a complete transaction?**	**Self-contained?**	**Leaves the business of the application being counted in a consistent state?**
	part of the Functional User Requirements.	Information alone is a complete transaction that is responsible to receive response from customer, record the response and return response to customer.	Information can be performed independently.	business need is achieved when the customer approves the reservation information and notifies the system, the system records the reservation and returns notification to the customer.
Assign Driver	Yes. Assign Driver is part of the Functional User Requirements.	Yes. Assign Driver is a complete transaction that is responsible to receive notification of driver assignment, record driver assignment and notify the driver.	Yes. Assign Driver can be performed independently.	Yes. The complete business need is achieved when a driver is assigned.
Pickup Customer	Yes. Pickup Customer is part of the Functional User Requirements.	No. Pickup Customer alone isn´t a complete transaction because the Reservation System must also record the reservation as completed in the Reservation database. These steps cannot be logically separated.	No. Finalize Reservation Request is a subsequent step that must be performed in order to complete the elementary process.	No. The complete business need is only achieved when the driver communicates to Reservation system that the customer was attended, and the Reservation System records the reservation request completion.
Finalize Reservation Request	Yes. Finalize Reservation Request is part of the Functional User Requirements.	No. Finalize Reservation Request alone isn´t a complete transaction because the driver must also communicate to	No. Pickup Customer is a prior step that must be performed in order to complete the elementary	No. The complete business need is only achieved when the driver communicates to Reservation

	Elementary Process Identification Rules			
Business Process Activities	**Meaningful to the user?**	**Constitutes a complete transaction?**	**Self-contained?**	**Leaves the business of the application being counted in a consistent state?**
		reservation system that the customer was picked up. These steps cannot be logically separated.	process.	system that the customer was attended, and the Reservation System records the reservation request completion.

The table below lists the EP identified by the analysis of how each business process activity attends the rules to identify an Elementary Process:

Business Process Activities	Identify an elementary process for each unit of activity identified that meets all of the EP identification rules
• **Request Reservation** • **Receive Reservation Request** • **Check Availability** • **Get Alternative Time** • **Provide Reservation Information**	Individually these activities do not meet all EP identification rules (see table above). They only satisfy all EP identification rules and fulfill the complete business need when combined. The EP is composed of the processing logic of all these business process activities.
• **Process Reservation**	This business process activity meets all EP identification rules in itself (see table above) and constitutes an EP.
• **Assign Driver**	This business process activity meets all EP identification rules in itself (see table above) and constitutes an EP.
• **Pickup Customer** • **Finalize Reservation Request**	Individually these activities do not meet all EP identification rules (see table above). They only satisfy all EP identification rules and fulfill the complete business need when combined. The EP is composed of the processing logic of all these business process activities.

Example 2: Diving School Application - Development Project

General description

A diving school requires a system for managing their information: contractors, facilities, working shifts to effectively manage the coverage of shifts by the instructors that are available at the diving facilities and aboard several boats for touristic dives and diving courses.

Functional User Requirements

FUR01

To handle the documentation for the insurance policies of the instructors and to be compliant with the current legislation, the school needs to store the following information for each Contractor:

- unique serial number
- first name and last name,
- address of residence,
- town of residence,
- postal code of residence,
- telephone number,
- whether in possession or not of nautical license.

To keep track of the 'career' of each Contractor, the school decided to associate with them a category, which can take the following values:

1. "Dive-master",
2. "Assistant Instructor",
3. "Instructor".

These values are fixed and are not expected to change over time; they can be assigned to each Contractor sequentially and represent the progress of 'careers' within the school (e.g. a new employee starts as "Dive-master" and advances over time to become "Instructor").

Forms must be created for entry, display, editing, and deletion of the Contractor's data.

To select the occurrence of data to edit, delete or view in detail, there will be an independent list box of items without accessory details: Serial number, Name, and Surname.

A function key will activate all the functions, and eventually generate an error/outcome message.

The deletion of a Contractor is only a logical one: no data will be physically deleted but will be labeled as obsolete. A Contractor cannot be deleted as long as there are working shifts associated with it.

FUR02

The school also needs to manage the Facilities ("diving", "boat" or "rubber dinghy - RD"), to each of which a unique friendly name is associated (for example: “diving_moneglia”, “diving_palau”, “blue-arrow-boat”, “goletta-boat”, “Genova_RD” and so on).

For each type of facility, the following information must be stored:

- unique identifier of the Facility
- description
- type
- number of people it can host
- number of available cylinders
- presence or absence of portable toilet
- presence or absence of drinking water reserve.

Forms must be created for entry, display, editing, and deletion of Facility data; a Facility cannot be deleted if there are active shifts.

In order to select the occurrence of data to edit, delete or view in detail, there will be an independent display of the list of items without accessory details: Identifier, Description, Type.

A function key will activate all the functions, and eventually generate an error/outcome message.

FUR03

Lastly, to effectively manage shifts coverage, the school needs to handle the following shift availability information of Contractors:

- unique identifier of the working shift,
- the serial number of the Contractor who is available (using a combo-box),
- the date on which availability was provided,
- the start date of the availability period,
- the end date of the availability period,
- preferred facility (using a combo-box)
- status (initially set to "tentative shift").

Forms must be created for entry, display, editing, and deletion of Shift availability data.

In order to select the occurrence of data to edit, delete or view in detail, there will be an independent display of the list of items without accessory details: Availability identifier, Contractor serial number.

A function key will activate all the functions, and eventually generate an error/outcome message.

FUR04

Each Contractor can provide more than one availability, each one initially in "tentative shift" status.

When consolidating an availability within a "shift", the Secretariat selects the desired availability and transform it using a specific command in an "assigned shift"; he/she can change the start and end dates of the diving period, and can assign the status of "operating shift" to the availability. To delete an operating shift after consolidation, the same function to delete a tentative shift can be used.

FUR05

The following queries are defined and required to:

1. Find out who, among the Contractors, already has a license and is, therefore, able to handle excursions at sea as "boatman" – attributes displayed: serial number, first name and last name, total.
2. Select all availabilities of Contractors received for any month of the current year (or other month and year) – attributes displayed: serial number, first name and last name, category, start date and end date of the availability period, and facility preference.
3. Calculate which is the maximum number of people managed by the school (considering both the dives and the boats), depending on the number and type of facilities in the archive.
4. Calculate the number of Contractors in each category (Dive-master; Assistant-instructor; Instructor) present in the archive – total and subtotal by category.
5. Select, by displaying first name and surname, the most "loyal" Contractors, i.e. the first three that have provided more availability since the beginning of the current year.
6. An enhanced version of query 4 (above), parametric by category – in other words, the capability of selecting a single category to display the list of Contractors (First name and last name) in addition to their total.

Solution

Identify the Boundaries of the Applications Involved in the Measurement

We may identify only one Software Application named Diving School.

Determine the measurement Purpose, Scope, and Type

The purpose of the measurement is to size the software produced by the development project and the baseline of the application after the initialization.

The scope of the measurement includes all the functionalities needed by the users to set up the first installation.

The type of measurement is “Development” and “Baseline after the first Development”.

Identify the IFPUG SFP BFCs

List the EP type Elements FUR	Function Type	Description	Operation Type	SFP tot
Transactions				
Contractor Management				
Enter Contractor	EP	Elementary Process	ADD	4.6
Edit Contractor	EP	Elementary Process	ADD	4.6
Display Contractor	EP	Elementary Process	ADD	4.6
Delete Contractor	EP	Elementary Process	ADD	4.6
Contractor choice list	EP	Elementary Process	ADD	4.6
Facility Management				
Add Facility	EP	Elementary Process	ADD	4.6
Edit Facility	EP	Elementary Process	ADD	4.6
Display Facility	EP	Elementary Process	ADD	4.6
Delete Facility	EP	Elementary Process	ADD	4.6
List for choice of Facility	EP	Elementary Process	ADD	4.6
Shift Management				
Enter Shifts	EP	Elementary Process	ADD	4.6
Edit Shifts	EP	Elementary Process	ADD	4.6
Display Shifts	EP	Elementary Process	ADD	4.6
Delete Shifts	EP	Elementary Process	ADD	4.6
Favorite Facility Combo-box	EP	Elementary Process	ADD	4.6
Choice of available Contractor number Combo-box	EP	Elementary Process	ADD	4.6
Assign Shifts				
Shift selection list	EP	Elementary Process	ADD	4.6
Assign shift	EP	Elementary Process	ADD	4.6
Queries				
List of Contractors with nautical licence	EP	Elementary Process	ADD	4.6
List of availability of contractors by period	EP	Elementary Process	ADD	4.6
Calculation of maximum number of people manageable by the school	EP	Elementary Process	ADD	4.6

List the EP type Elements FUR	Function Type	Description	Operation Type	SFP tot
Statistics of Contractors present in archive by category	EP	Elementary Process	ADD	4.6
Statistics of most habitial Contractors	EP	Elementary Process	ADD	4.6
Statistics of Contractors present in archive with choice of category	EP	Elementary Process	ADD	4.6
Total Eps				**110.4**

List the LF type Elements

FUR	Function Type	Description	Operation Type	SFP tot
Logical Files				
Contractor	LF	Logical File	ADD	7.0
Facility	LF	Logical File	ADD	7.0
Shift	LF	Logical File	ADD	7.0
Total LFs				**21.0**

Calculate the functional size

The size of the software produced by the development project is obtained using the following formula:

DSFP = ADD + CFP

So adding the EP component to the LF component and considering that there are no conversion's functionalities, we have:

DSFP = (110.4 + 21.0) + 0 = 131.4 SFP

The baseline count after development will be:

ASFP = ADD

ASFP = 131.4 SFP

Example 3: Diving School - Enhancement Project

General description

With reference to the previously measured Diving School system, an Enhancement Project for the Software is proposed.

Functional User Requirements

FUR01

The user wants to eliminate the Contractor's delete function.

FUR02

The presence/absence of a ship doctor during excursions must be managed in the file DIVING FACILITIES.

FUR03

For tax and safety reasons the function to delete availability shifts will no longer be required.

FUR04

The user also needs to manage the information of course Participants and their enrollment in the excursion. The information managed will be: Participant ID, LastName, FirstName, Date of Birth, diving license, license date.

To enroll in the excursion shift the following information is required: Participant ID, Shift ID, excursion date, duration, final examination passed (YES/NO).

Selection of the participants in an excursion shift will be obtained by List Box on the Participant file, containing Participant ID, FirstName, LastName.

For Shift selection, the List box already available in the basic application will be used.

A function key will activate the function, and eventually generate an error/outcome message.

A functionality will be provided to populate initially the participant's file with an already existent archive of past participants.

FUR05

The user also needs to be able to issue at the end of the course a certificate of attendance to all participants enrolled in an excursion.

The information to manage, in addition to the participant master data, are the excursion shift the participant is enrolled in, the date of the excursion, the duration, the name of the instructor, and the doctor if present.

For Shift selection, the List box already available in the basic application will be used.

A function key will activate the functions, and eventually generate an error/outcome message.

FUR06

The user also needs to be able to issue a Certificate of participation as an instructor to Contractors, in which the information will be, in addition to master data: Shift ID, Instructor ID, excursion date, ship doctor if present.

For Shift selection, the List box already available in the application will be used.

A function key will activate the functions, and eventually generate an error/outcome message.

Solution

Identify the Boundaries of the Applications Involved in the Measurement

We may identify only one Software Application named Diving School to be enhanced.

Determine the measurement Purpose, Scope, and Type

The purpose of the measurement is to size the changes in the software application needed by the users and the size of the baseline of the application after the changes.

The scope of the measurement for the enhancement project includes all the functionalities "worked" by the enhancement project (added, changed, and deleted).

The scope of the measurement of the baseline is to size the whole application after the enhancement project.

The type of measurement is "Enhancement" and "Baseline after Enhancement.

Identify the IFPUG SFP BFCs

List the EP type Elements

FUR	Function Type	Description	Operation Type	SFP tot
Transactions				
Contractor Management				
Delete Contractor	EP	Elementary Process	DEL	4.6
Facility Management				
Add Facility	EP	Elementary Process	CHG	4.6
Edit Facility	EP	Elementary Process	CHG	4.6
Display Facility	EP	Elementary Process	CHG	4.6
Delete Facility	EP	Elementary Process	DEL	4.6
Assign Shifts				
Enrollment Management				
Enter Enrollment	EP	Elementary Process	ADD	4.6
Edit Enrollments	EP	Elementary Process	ADD	4.6
Display Enrollments	EP	Elementary Process	ADD	4.6
Delete Enrollments	EP	Elementary Process	ADD	4.6
Queries				
Generate Certificate of Attendance	EP	Elementary Process	ADD	4.6
Generate Contractor participation	EP	Elementary Process	ADD	4.6
Participant Management				
Add Participant	EP	Elementary Process	ADD	4.6

FUR	Function Type	Description	Operation Type	SFP tot
Edit Participant	EP	Elementary Process	ADD	4.6
Display Participant	EP	Elementary Process	ADD	4.6
Delete Participant	EP	Elementary Process	ADD	4.6
Participant Combo-box	EP	Elementary Process	ADD	4.6
Initial population of the Participant's LF	EP	Elementary Process	CFP	4.6
Total Eps				**78.2**

List the LF type Elements

FUR	Function Type	Description	Operation Type	SFP tot
Logical Files				
Facility	LF	Logical File	CHG	7.0
Participant	LF	Logical File	ADD	7.0
Enrollment in shift	LF	Logical File	ADD	7.0
Total LFs				**21.0**

Calculate the functional size

The size of the software produced by the enhancement project is obtained using the following formula:

ESFP = ADD+CHG+DEL+CFP

Summarizing the different component, we have:

	SFP
ADD	64.6
CHG	20.8
DEL	9.2
CFP	4.6

ESFP = 64.6 + 20.8 + 9.2 + 4.6 = 99.2 SFP

The baseline count after development will be:

ASFPA = ASFPB + ADD – DEL

ASFPA = 131.4 + 64.6 – 9.2 = 186.8 SFP

Example 4: Tickets Sale - Development Project

General description

The "Ticket Sales (TS)" application handles on-line ticket sales for entertainment events. It has a final user interface and a different interface for travel agency operators throughout the country. All administrative and accounting tasks are handled by a separate "Back Office (BO)" application. The two systems interact by exchanging appropriate messages. For example, the BO issues notifications concerning updating of the show calendar file and receives lists of tickets sold and to ship (see next section FUR03).

Functional User Requirements

FUR01

After selecting an event the user purchases a ticket and confirms it after entering the payment.

FUR02

Payment through the travel agency operator can be made in cash or by credit card (in the latter case user authentication is required; if not registered, the user can register and manage its profile)

FUR03

After payment, the ticket can be printed. The printed ticket includes the event name, name, and surname of the purchaser, price, seats, and the code of the travel agency issuing the ticket.

When using the system from home, the customer can enter the address of the place where they prefer the ticket to be sent.

The supporting documentation is as follows:

a) the list of the application data
b) a use case diagram of the system
c) a use case diagram of ticket purchase

List of application data:

1. Calendar (EventID, Show Type, Title, Description, Ticket Price, location, Start date)
2. Tickets (EventID, FirstName, LastName, TaxID, Seats, Purchase Date, Payment Type, Total Payment, CardNumber (if Payment Type = credit card), Security Code (if Payment Type = credit card), Delivery, Shipping Address - Street, City, Province, Region - (if Delivery = shipment)
3. Seats available (EventID, Capacity, Seats available)
4. Registered users master data (10 attributes)
5. Credit cards (FirstName, LastName, TaxID, CardNumber, Type, ExpirationDate, Status (active/blocked))
6. Agencies (5 attributes)

System Use Case:

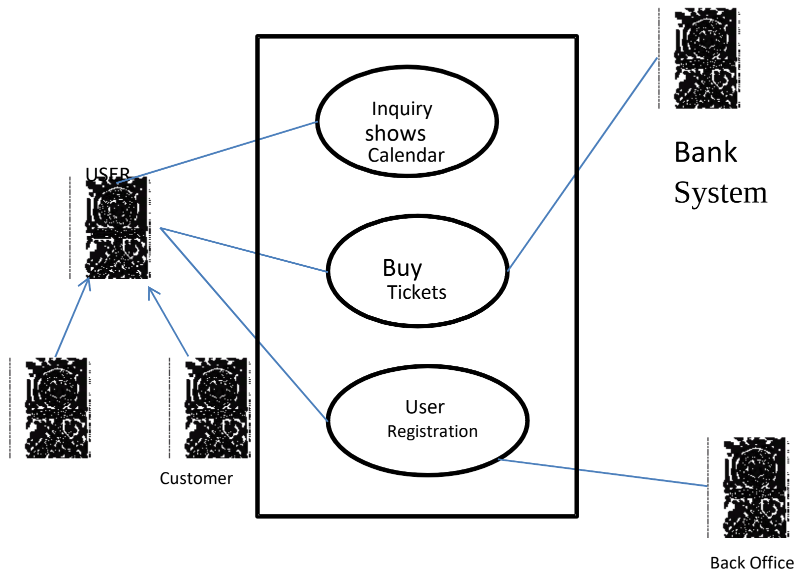


Figure 4 – Example 4 System Use Case

Ticket Sales Use Cases

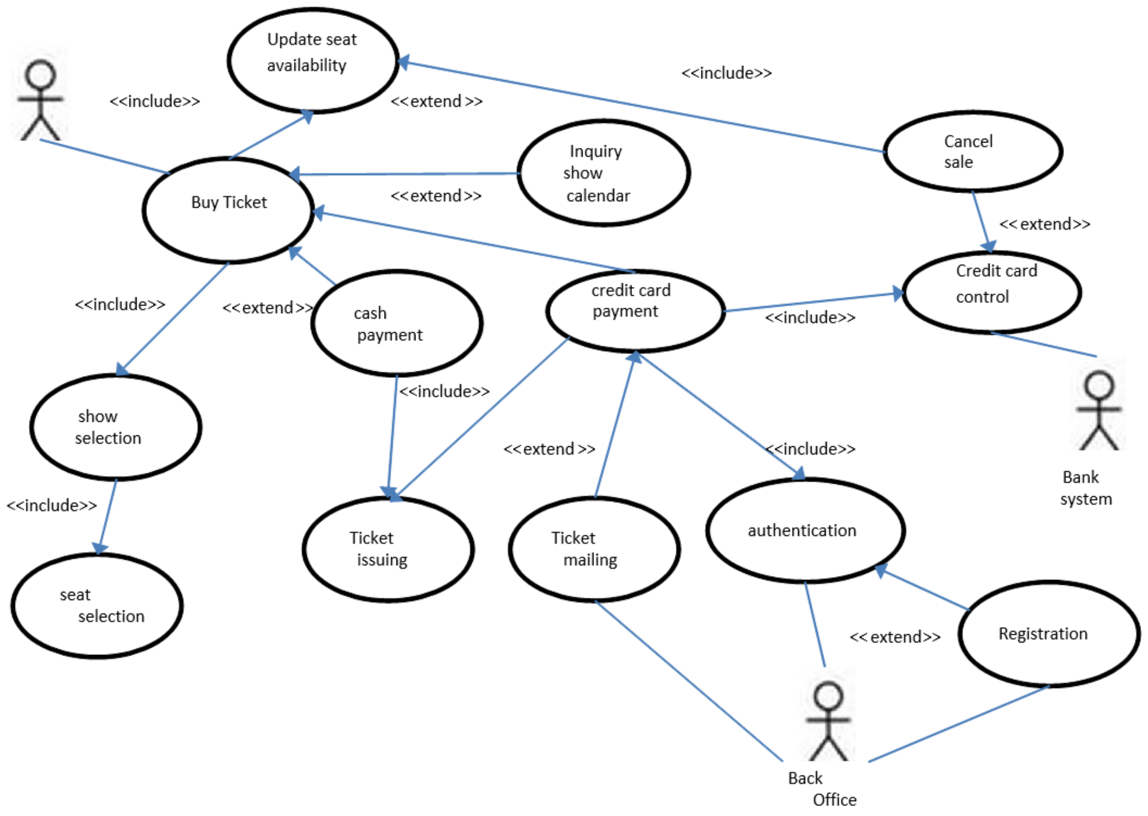


Figure 5 – Ticket Purchase Use Case

Solution

Identify the Boundaries of the Applications Involved in the Measurement

We may identify two Software Applications named “Ticket Sales (TS)” and “Back Office (BO)". Both of them are seen from a business perspective and the information exchanges are on a “peer to peer” basis. No technological aspects were taken into consideration in identifying boundaries. We will count here only one of the two applications: the Tickets Sale.

Determine the measurement Purpose, Scope, and Type

The purpose of the measurement is to size the software produced by the development project and the baseline of the application after the initialization.

The scope of the measurement includes all the functionalities needed by the users to set up a first installation.

The type of measurement is “Development” and “Baseline after the first Development”.

Identify the IFPUG SFP BFCs

List the EP type Elements

FUR	Function Type	Description	Operation Type	SFP tot
Transactions				
User Profile				
User enrollment	EP	Elementary Process	ADD	4.6
Edit User	EP	Elementary Process	ADD	4.6
Delete User	EP	Elementary Process	ADD	4.6
Display User	EP	Elementary Process	ADD	4.6
Display Calendar	EP	Elementary Process	ADD	4.6
Purchase Ticket	EP	Elementary Process	ADD	4.6
Ticket issue	EP	Elementary Process	ADD	4.6
Tickets sold sent to BO	EP	Elementary Process	ADD	4.6
Tickets to be shipped sent to BO	EP	Elementary Process	ADD	4.6
Total Eps				**41.4**

List the LF type Elements

FUR	Function Type	Description	Operation Type	SFP tot
Logical Files				
Registered users master data	LF	Logical File	ADD	7.0
Calendar (includes av. seats)	LF	Logical File	ADD	7.0
Ticket	LF	Logical File	ADD	7.0
Credit card	LF	Logical File	ADD	7.0
Agency	LF	Logical File	ADD	7.0
Total LFs				**35.0**

Calculate the functional size

The size of the software produced by the development project is obtained using the following formula:

DSFP = ADD + CFP

So adding the EP component to the LF component and considering that there are no conversion’s functionalities, we have:

DSFP = (41.4 + 35.0) + 0 = 76.4 SFP

The baseline count after development will be:

ASFP = ADD

ASFP = 76.4 SFP

Example 5: Tickets Sale - Enhancement Project

General description

With reference to the previously measured Ticket Sale system, an Enhancement Project for the Software is proposed.

Functional User Requirements

FUR01

Statistics about the most sold tickets by Show type must be provided.

FUR02

Every week all ticket purchases must be sent to the accounting system.

FUR03

A new function to cancel all tickets expired at a certain date must be provided.

FUR04

A function to delete all registered users by selecting the last ticket purchase date must be provided.

FUR05

There is also the requirement to add the logo of the event in the ticket printing.

Solution

Identify the Boundaries of the Applications Involved in the Measurement

We may identify only one Software Application named Tickets Sale to be enhanced.

Determine the measurement Purpose, Scope, and Type

The purpose of the measurement is to size the changes in the software application needed by the users and the size of the baseline of the application after the changes.

The scope of the measurement for the enhancement project includes all the functionalities "worked" by the enhancement project (added, changed, and deleted).

The scope of the measurement of the baseline is to size the whole application after the enhancement project.

The type of measurement is "Enhancement" and "Baseline after Enhancement.

Identify the IFPUG SFP BFCs

List the EP type Elements

FUR	Function Type	Description	Operation Type	SFP tot
Transactions				
Statistics of the most sold tickets by Show type	EP	Elementary Process	ADD	4.6
All tickets purchased in the period sent to the accounting system	EP	Elementary Process	ADD	4.6
Tickets expired at a certain date deleted	EP	Elementary Process	ADD	4.6
Registered users by ticket purchase date deleted	EP	Elementary Process	ADD	4.6
Total Eps				**18.4**

List the LF type Elements

FUR	Function Type	Description	Operation Type	SFP tot
Logical Files				
Registered users master data	LF	Logical File	CHG	7.0
Ticket	LF	Logical File	CHG	7.0
Total LFs				**14.0**

Calculate the functional size

The size of the software produced by the enhancement project is obtained using the following formula:

ESFP = ADD+CHG+DEL+CFP

Summarizing the different component, we have:

	SFP
ADD	18.4
CHG	14.0
DEL	0
CFP	0

ESFP = 18.4 + 14.0 + 0 + 0 = 32.4 SFP

The baseline count after development will be:

ASFPA = ASFPB + ADD – DEL

ASFPA = 76.4 + 18.4 – 0 = 94.8 SFP

Part 3 – Appendices

This page intentionally left blank

Part 3 - Appendices

Contents

Part 3 includes the following sections:

This page intentionally left blank

Part 3 Appendix A

Glossary

This is a comprehensive glossary of terms used in this manual

Adaptive Maintenance (ISO/IEC 14764:2006): The modification of a software product, performed after delivery, to keep a software product usable in a changed or changing environment. Adaptive maintenance provides enhancements necessary to accommodate changes in the environment in which a software product must operate. These changes are those that must be made to keep pace with the changing environment. For example, the operating system might be upgraded and some changes may be made to accommodate the new operating system.

Application: A cohesive collection of automated procedures and data supporting a business objective; it consists of one or more components, modules, or subsystems.

BFC (ISO/IEC 14143-1:2007 (R2019)): Base Functional Component – Elementary unit of Functional User Requirements defined by and used by an FSM Method for measurement purposes.

Boundary: A conceptual interface between the software under study and its users.

Business data: May also be referred to as Core User Data or Business Objects. This type of data reflects the information needed to be stored and retrieved by the functional area addressed by the application. Business Data usually represents a significant percentage of the entities identified.

Code data: The user does not always directly specify Code Data, sometimes referred to as List Data or Translation Data. In other cases, it is identified by the developer in response to one or more technical requirements of the user. Code Data provides a list of valid values that a descriptive attribute may have. Typically, the attributes of the Code Data are Code, Description, and/or other 'standard' attribute describing the code; e.g., standard abbreviation, effective date, determination date, audit trail data, etc.

Computational purpose: It is the goal that characterizes the set of functional requirements that identifies an EP: to move and process information about objects of interest to and from the user.

Consistent state: The point at which processing has been fully executed; the Functional User Requirement has been satisfied and there is nothing more to be done. An EP drives the system (application) from a consistent state into another consistent state.

Control Information: Data that influences an elementary process by specifying what, when, or how data is to be processed.

Conversion Functionality: Transactional or data functions provided to convert data and/or provide other user-specified conversion requirements.

Corrective maintenance (ISO/IEC 14764:2006): The reactive modification of a software product performed after delivery to correct discovered problems. The modification repairs the software product to satisfy requirements.

Development project functional size: A measure of the functionality provided to the users with the first release of the software, as measured by the development project function point count by the activity of applying the IFPUG Simple Function Point Measurement Method.

Elementary process (EP): The smallest unit of activity that is meaningful to the user, that constitutes a complete transaction, it is self-contained and leaves the business of the application being measured in a consistent state.

Enhancement project functional size: The enhancement project functional size is a measure of the functionality added, changed, or deleted at the completion of an enhancement project, as measured by the enhancement project function point count by the activity of applying the IFPUG Simple Function Point Measurement Method.

Entity: A fundamental thing of relevance to the user, about which a collection of facts is kept. An association between entities that contain attributes is itself an entity.

Functional Size (ISO 14143-1:2007 (R2019)): Size of the software derived by quantifying the Functional User Requirements.

Functional User Requirements (FUR) (ISO 14143-1:2007 (R2019)): A subset of User Requirements. FURs are user practices and procedures that the software must perform to meet the user requirements. They do not include quality, performance, or technical requirements.

IFPUG: International Function Point Users Group

Logical File: A Logical File represents functionality provided to the user to meet internal and external data storage requirements.

Measurement: The task of measuring and its result, assigning a value to an attribute in accordance with a scale of reference.

Measurement method: A logical sequence of operations performed to produce measurements

Non-Functional User Requirements (NFR) (ISO 14143-1:2007): A software requirement that describes not what the software will do but how the software will do it.

Perfective Maintenance (ISO/IEC 14764:2006): Modification of a software product after delivery to detect and correct latent faults in the software product before they are manifested as failures. Perfective maintenance provides enhancements for users, improvement of program documentation, and recoding to improve software performance, maintainability, or other software attributes. Contrast with adaptive maintenance; corrective maintenance.

Preventive Maintenance (ISO/IEC 14764:2006): the modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults.

Processing logic: Any of the requirements specifically requested by the user to complete an elementary process such as validations, algorithms or calculations, and reading or maintaining a data function.

Project Requirements and Constraints: Requirements that define how a software system project should be managed and resourced, or constraints that affect its performance.

Project Requirements may include:

- the targets the project should achieve (e.g. budget, delivery date, product quality);
- the project management processes that should be used;
- how the project should be governed and resourced.

Constraints may include:

- limitations on the project resources planned or needed;
- dependencies on other projects outside the control of the project concerned.

Quality Requirements Any requirements relating to software quality as defined in ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

Scope of Measurement: The scope of measurement defines the user features that will be included in a specific IFPUG Simple Function Points measurement. The scope:

- defines a (sub)set of the software to be measured;
- is determined by the goal of for the measurement activity;
- identifies which user functions should be included in the measurement for providing the required answers of the measurement goal;
- it may include more than one software application.

Self-contained: No prior or subsequent processing steps are needed to initiate or complete the Functional User Requirement(s).

SFP: Simple Function Points.

SNAP: Software Non-Functional Assessment Process

SPM: Simple function point counting Practices Manual

Storage purpose: It is the goal that characterizes the set of functional requirements that identifies an LF: to preserve logically related information about objects of interest to the user.

Technical requirements: Requirements related to the technology and environment, for the development, maintenance, support, and execution of the software. They belong to the Non-Functional Requirement category in the ISO 14143-1:2007 (R2019) standard.

User: Any person or object that communicates or interacts with the software at any time**.**

User recognizable: The term user recognizable refers to requirements for processes and/or data that are agreed upon, and understood by, both the user(s) and software developer(s).

User Requirements: Requirements describing what the user is asking for. (UR)

User view (Functional) A user view is the Functional User Requirements as perceived by the user.

Value: The numeric value assigned to an attribute of an entity as a result of a measurement.

Part 3 Appendix B

ISO/IEC/IEEE – Definitions

User Requirements

In 1998, the first ISO/IEC Functional Size Measurement standard was published (ISO/IEC 14143-1:1998 "Software and Systems Engineering – Software measurement – Functional size measurement – Definition of concepts"). This standard defines the Functional Size as "a size of the software derived by quantifying the Functional User Requirements" (FUR). (This standard was updated in 2007 and is currently published as ISO/IEC 14143-1:2007 (R2019).)

ISO/IEC 14143-1 distinguishes two subsets of user requirements (UR):

Functional User Requirements

Non-Functional Requirements (NFR)

ISO/IEC 9126-1:2001 provided the definition of the characteristics and associated quality evaluation process to be used when specifying the requirements for and evaluating the quality of software products throughout their life cycle.

ISO/IEC 25010:2011 has replaced and improved ISO/IEC 9126-1

ISO/IEC 14143-1 definition of FUR is as follows:

Functional User Requirements

"A subset of the User Requirements (UR). Requirements that describe what the software shall do, in terms of tasks and services."

Note: Functional User Requirements include but are not limited to:

- data transfer (for example, Input customer data, Send control signal)
- data transformation (for example, Calculate bank interest, Derive average temperature)
- data storage (for example, Store customer order, Record ambient temperature over time)
- data retrieval (for example, List current employees, Retrieve aircraft position)

Examples of User Requirements that are not Functional User Requirements include but are not limited to:

- quality constraints (for example, usability, reliability, efficiency, and portability)
- organizational constraints (for example, locations for operation, target hardware, and compliance to standards)
- environmental constraints (for example, interoperability, security, privacy, and safety)
- implementation constraints (for example, development language, delivery schedule)

ISO/IEC 14143-1 does not define Non-Functional Requirements themselves but gives some examples in a note after FUR definition. In 2009, a separate initiative under development (ref. ISO/IEC/IEEE 24765:2010 Systems and software engineering – Vocabulary) proposed a formal definition of a Non-functional User Requirement, as follows:

Non-Functional User Requirements (NFR)

- A software requirement that describes not what the software will do but how the software will do it. [ISO/IEC 24765, Systems and Software Engineering Vocabulary.] Syn: design constraints, non-functional requirement. See also: functional requirement.

Part 3 Appendix C

Index