Software Non-functional Assessment Process (SNAP)

Assessment Practices Manual

Release 2.2

This page intentionally left blank

International Function Point Users Group (IFPUG)

SNAP Assessment Practices Manual

Release 2.2

Chairperson, Non-Functional Sizing Standards Committee (NFSSC)
Talmon Ben-Cnaan
NFSSC@ifpug.org

ISBN 978-0-9830330-9-7

This page intentionally left blank

Release 2.2, April 2014

This release replaces Release 2.1, which is now obsolete.
Changes are made periodically to the information within.

Documentation Team

NFSS Committee

Christine Green, IFPUG Director, Board Liaison

Talmon Ben-Cnaan, Amdocs (Chairperson)
Kathy Lamoureaux Aetna (Vice Chairperson)

Abinash Sahoo, Amdocs; Roopali Thapar, IBM; Steve Chizar, Retired, Naval Systems Supply Business Solutions; Charley Tichenor, DCA; Jalaja Venkat, iGATE; Luigi Buglione, Engineering.IT; Mauricio Aguiar, TI Metricas

SNAP Assessment Practices Manual Team 2011-2012

Christine Green, IFPUG Director, Board Liaison

Steve Chizar, Retired, Naval Systems Supply Business Solutions, Chair

Abinash Sahoo, Amdocs, Vice Chairperson

Charlene Zhao, BMO; Charley Tichenor, DSCA; Jalaja Venkat, iGATE; Joanna Soles, CSC; Kathy Lamoureaux, AETNA; Peter R Hill, ISBSG; Roopali Thapar, IBM; Talmon Ben-Cnaan, Amdocs

IT Performance Committee

Dan Bradley, Chair

Christine Green, HP Enterprise Services, SNAP Project Manager

Wendy Bloomfield, Great-West Life Assurance Co.

Talmon Ben-Cnaan, Amdocs

Joanna Soles, CSC

Loredana Fralliccіardi, CSC, IFPUG Director, SNAP Sponsor (2008-2010)

Janet Russac, Software Measurement Expertise, Inc., IFPUG Director, SNAP Sponsor (2010-2011)

SNAP Assessment Practices Manual Team 2009-2011

Patty Boyce, NASCO; Sergio Brigido, HP Enterprise Services; Steve Chizar, Retired, Naval Systems Supply Business Solutions; Sheila Dennis, The David Consulting Group; David Garmus, The David Consulting Group; Asha Goyal, Nucleus Software Exports Ltd; Claudia Hazan, Employee of a Brazilian Gov IT Organization; Peter Hill, ISBSG; Lori Holmes, Q/P Management Group, Inc.; Ashwin Krishnamurthy, IBM India Pvt Ltd; Monica Lelli, IT Consultant; Luca Santillo, Agile Metrics; Jalaja Venkat, iGATE; Charles Wesolowski, QinetiQ North America Systems Engineering Group

SNAP Framework Team 2008

Albert Meyn, W&W Informatik GmbH; Ana Miccolis Petrobras; Ashwin Krishnamurthy, IBM; Carolyn Melvin, Northrop Grumman; Charles Wesolowski, Lockheed Martin Space Systems Company; David Seaver, PRICE Systems LLC; David Garmus, The David Consulting Group; Graeme Prescott, Australian Government; Jalaja iGATE Lori Holmes, Q/P Management Group, Inc.; Marco Motta, Vodafone; Mick Burn-Murdoch, Lalcrest Ltd; Mike Pearl, EDS; Muhammet Oeztuerk, W&W Informatik GmbH; Peter Grüner, G4Q IT Services; Pranabendu Bhattacharyya, Tata Consultancy Services

Special Thanks

A special thanks is offered to the team members of the two projects that have worked on defining the SNAP Framework as well as defining and creating the Assessment Practices Manual (APM), including:

- The two teams
- All of the individuals and organizations participating in review, beta test, and feedback
- Bonnie Brown, and Luc Vangrunderbeeck, for their support with the APM layout, the beta test tool, and the analysis of beta test data.

SNAP Reviewers – Thank You!

Ajith PN, Alain Gontard, Albert Meyn, Alcione Jandir Candéas Ramos, Amith Kulkarni, Anil Joshi, Atluri Kumar, Bonnie Brown, Candéas Ramos, Checng Cheng Zhao, Connie Smith, Dan Schultz, Dimple Gupta, Elisa Gallo, Emilia Leboreiro, Ganeshan Guruswamy, Giselle Fróes, Gurpreetkaur Marve, Guruprasath Sethumadhavan, Jay Fischer, Jaya Venkat, Jeff Lindskoog, Jim Watson, Joel Andre, Julian Gómez Bejarano, Jyoti Namjoshi, Kareem Quereshi, Kathy Lamoureaux, Krupanandra Babu, Krzysztof Kłosowski, Lionel Perrot, Luigi Buglione, Luis Andasola, Luis Carlos Grande, Luís Geminianno, Luz Ibáñez, Manoj Ramchandra Sable, Marcelo Leme, Marcin Soboń, Mario Vacalebri, Michael Fuller, Muhammet Oezteurk, Namit Mishra, Neetika Jain, Nicolas Chantereau, Nitin S Dharrao, Pablo Soneira, Padmanabhan Kalyanasundaram, Parag Saha, Peter Grüner, Peter Thomas, Philippe Perrin, Piotr Popowski, Pranabendu Bhattacharyya, Radhika Srinivas, Rafał Wasilewski, Rajib Banerjee, Ranjana Jain, Ravindra Moghe, Renata Faray, Rocío Juárez, Sachin Thakur, Savitha G, Sherry Ferrel, Shreyas Bhargave, Siddharth Misra, Sridhar Maheswaram, Srividhya Ganesan, Stanley Richardson, Susan Walls, Sushmitha Anantha ,Swapnil Dambe, Tomasz Gąsiorowski, Tomasz Marchel, Usha Yegnaseshan, Valkal Bairagi, Viral Shah, Wojciech Dratnal

SNAP Beta test and review companies – Thank You!

Accenture, ALLIGRA (groupe Astek), AMDOCS, Banco Bradesco S/A, Banco do Brasil , Banco de Crédito del Perú, BRQ IT Solutions, Business Integration Partners S.p.a., Centurylink , Cognizant Technology Solutions, CSI Piemonte, Engineering.IT SpA , FÓTON INFORMÁTICA S.A, GE Energy, GDF Suez , HP Enterprise Services, IBM , Kubota Systems Inc., Level 4 Ventures, Inc., MindTree, Mphasis an HP Company, Neoris, Orange Polska, SEMANTYS (Groupe Astek), SIDI srl, Sopra Group, Tata Consultancy Services Ltd, TELUS, TI METRICAS SERVICOS LTDA, T-Systems, Austria,UST global

For information about additional copies of this manual, contact
IFPUG
191 Clarksville Road
Princeton Junction, NJ 08550
U.S.A.
(609) 799-4900
E-mail: **ifpug@ifpug.org**
Web: http://www.ifpug.org

Table of Contents

This page intentionally left blank

Preface

Introduction

Having both functional size and non-functional size provides a more complete picture of software development. The functional size is quantifiable and represents a good measure of the functional project/application size. Providing a quantifiable measure for the Non-Functional Requirements (NFR) allows organizations to build historical data repositories that can be referenced to assist in decision making for the technical and/or quality aspects of applications.

A non-functional assessment will assist Information Technology (IT) organizations in multiple ways. It will provide insight into projects and applications to assist in estimating and in the analysis of quality and productivity. Used in conjunction with function point analysis, the non-functional assessment provides information that can identify items impacting quality and productivity in a positive or negative way. Having this information enables software professionals to:

- better plan, schedule and estimate projects
- identify areas of process improvement
- assist in determining future technical strategies
- quantify the impacts of the current technical strategies
- provide specific data when communicating non-functional issues to various audiences

SNAP History & Background

The IT Performance Committee (ITPC), changed to be the IT Measurement Analysis Committee (ITMAC), received approval from the IFPUG Board to proceed with the Technical Size Framework (TSF) Project at the 2007 ISMA conference. The goal of the project was to define a framework that would size the technical aspects of software development. The current issue with functional size is that it has not been suitable for sizing the technical requirements associated with a software development project. The focus of the project was to develop a separate technical size framework from the function point (FP) methodology as defined by IFPUG. Making this separate from the functional size measure would ensure that historical function point data could continue to be used. The defined framework had to be agreed to and supported by the IFPUG Board and the IFPUG members. The final product would define guidelines and rules for sizing the non-functional aspects of software development and would include non-functional aspects.

Release Draft

The work of the Software Non-functional Assessment Process (SNAP) project team resulted in a first draft version released in October 2009 for the purpose of getting it reviewed by a team of reviewers as well as the IFPUG Board, The IFPUG New Environment Committee and the IFPUG Counting Practice Committee. This was reflected in Release 0.1 (October 2009) of the International Function Point Users Group (IFPUG) SNAP Assessment Practices Manual (APM).

Release Beta

The work of the SNAP project team, along with the feedback from the reviewers resulted in a first Beta version released in November 2010 for the purpose of getting it reviewed by the Software Industry in general. This was reflected in Release 1.0 BETA (November 2010) SNAP APM. The feedback from the first review resulted in changes to small parts of the APM.

A beta test was performed in January 2011 on the second Beta version to confirm the process as well as to provide data for consolidation of the model. Small changes to the APM were made as well as confirmation of the SNAP Sizing Units used for most sub-categories. A few changes were made to the sub-categories, including adding one new sub-category.

An additional Beta test was performed in May 2011 to get the final data for consolidation of the calculation of the model to a consistent size measure.

The beta test was performed by multiple companies from around the globe.

Release 1.0

The initial public release of the SNAP APM (September 2011) defined the concepts, processes, and rules for assessing non-functional software requirements. Release 1.0 of the APM resulted from years of work by the SNAP team and its expert contributors. This release included additions and corrections collected during the beta test period and additional input from the team.

Release 2.0

The second public release of the SNAP APM (January 2013) refines the concepts, processes, and rules for assessing non-functional software requirements.

More definitions have been added to clarify the terms used in this manual.

Some sub-categories were re-defined based on users' experience and comments.

Guidelines have been added to refine the linkage between FP and SNAP, and to detail how to size requirements that involve both functional and non-functional aspects. Users should verify that there is no duplicated sizing; SNAP is complementary to FPA and not a replacement.

Release 2.1

The third public release contains more clarifications, based on questions and comments of users.

In addition, the sizing of Coda Data was inserted into the relevant sub-categories.

More clarifications were added to the sub-categories:

- Sub-category 1.2 Logical and Mathematical Operations, to explain the term "extensive mathematical operations"
- Sub-category 1.5, the SCU definition now includes the case of several logical files (as shown in Example 4). Tables 2-6 and 2-7 were modified to better explain how SP calculation is done.
- Sub categories 2.3 and 2.4 Multiple Inputs and Output methods, sub-category 4.2 Multiple Input / Output interfaces: Clarifications were added to the complexity parameters to clarify that SP size derived from the number of additional methods or interfaces.
- Clarifications were added to Table 2-16 and 2-17 to the counting formula.

Release 2.2

This release contains additional clarifications and error corrections.

- Adding SNAP certification paragraph to the Introduction chapter.
- Removing "the batch function" and "single application" from the examples of partition.
- Adding brackets (ling to FPA) to Boundary and Partition Hints.
- Clarifications to the definition of "Logical file."
- Definition of "database view" was added.
- In sub-category 1.2, name was replaced from "FTR complexity" to "FTR density."
- A note was added to 1.2.
- Clarification to SOA was added to notes in sub-category1.4.
- Definition of Logical Files moved to "Terms."
- "Labels" added as a UI element to sub-category 2.1, and an example was added.
- Note added to 2.2 stating that calibration of this sub-category remains open for

future research.

- Calculation example in sub-category 3.1 was corrected.
- Example in sub-category 3.2 was corrected.
- The note dealing with code data was removed from 3.2.
- In figure 1-5, boundary was added. Header changed from hotel reservation to holiday reservation.
- A note was added to sub-category 4.2.
- Table 1-25 was fixed.
- Example 1, 6 and 9 were expanded and SP calculations were changed.
- Example 4, Tables 2-6 and 2-7 were changed (more accurate terms).
- Example 8 was fixed, to comply with the second note on page 5-30.
- Clarifications added to figure 2-1.
- In example 12, SP calculations were corrected.
- In table 3-2, for 2.2 Help Methods, the "what to check" field was corrected.
- Terms added to the glossary (ASPA, DET, DSP, and NFR). "Technical size framework" and "Data element type" removed from glossary.

Future Releases

This document is meant to be a living one. We must recognize how to perform assessments in new environments as they are introduced. We need to be able to do this in the context of maintaining the validity of the assessments 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 Non-Functional Standards Sizing Committee wishes to thank all those who have helped us in our research and in the production of this manual.

Talmon Ben-Cnaan

Chairperson, NFSSC.

Introduction to the Assessment Practices Manual

Introduction

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

Contents

This chapter includes the following sections:

Objectives of the Assessment Practices Manual

The primary objectives of the IFPUG Assessment Practices Manual are to

- Provide a clear and detailed description of Non-functional Assessment.
- Ensure that assessments are consistent.
- Provide guidance to allow non-functional assessment to be applied to popular methodologies and techniques.
- Provide a common understanding to allow tool vendors to provide automated support for non-functional assessment.

SNAP Beta Test

The purpose of the SNAP beta test was to repeat the spirit of Dr. Allan Albrecht's test of the initial version of the function point methodology, as documented in his 1977 paper "Measuring Application Development Productivity." Our beta test, similar to Dr. Albrecht's function point test, found a statistically significant correlation between SNAP count size and work effort using a statistically large sample size of 48 applications containing over 500 data entries. The r^2 for the correlation between SNAP count and work effort was .89, the Spearman rank correlation was .85, the corresponding p-values for both tests was below .0001, and the test for randomness in the regression model passed the runs test.

These statistics mean that for this beta test, SNAP size was 89% of the reason for the work effort expended (the other 11% may result from different software languages, teams skills, counting errors, etc.)

Documents Used

The following documentation was used to develop this release:

- "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 SNAP with current Functional Sizing principles.
- "Framework for Functional Sizing"; this IFPUG paper explains that product size contains three dimensions: functional size, technical size and quality size. The IFPUG FPA-method provides a measure for the functional size.
- ISO/IEC 14143-1:2007 "Information technology – Software measurement – Functional size measurement – Part 1: Definition of concepts" and ISO/IEC/IEEE 24765:2010 "Systems and software engineering – Vocabulary" (www.computer.org/sevocab), as standard references for Functional and Non-Functional Requirements definitions.

Issues not sufficiently covered in the sources listed above were decided by the IFPUG IT Performance Committee based on variants of existing counting practices and validated through impact studies.

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

Intended Audience

The standard in this manual should be applied by anyone to assess non-functional product size. The manual was designed for use by persons new to assessment as well as those with intermediate and advanced experience.

Organization of the Assessment Practices Manual

There are three major parts in the Assessment Practices Manual (APM)

- Part 1: The SNAP Method
- Part 2: Examples
- Part 3 Appendices

Part 1 – The SNAP Method

Part 1 is the IFPUG non-functional size measurement method which contains the rules. To speak a language as a native, learning the grammar and the words alone are not sufficient. They just provide a framework. You need language experience to understand how the language is spoken in practice, how the grammar rules should be applied, what idiomatic expressions are common, and so on. The same is true for SNAP. The knowledge of process and rules, as reflected in Part 1, is a necessity, but the knowledge alone is not a sufficient condition to apply SNAP correctly. That's why the APM contains the parts below.

Part 2 – Examples

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

Part 3 – Appendices

Part 3 contains additional information (Glossary, APM link to IFPUG CPM, Index).

In principle, each part stands on its own.

Manual Revision Process

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

Frequency of Changes

During January of each year, a new version of the Assessment Practices Manual may become effective. It will include any new or changed definitions, rules, or Assessment practices that have been finalized by the Non-Functional Sizing Standards Committee (NFSSC) since the previous version.

Change Process

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

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

1 Issue Submitted

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

2 Research Assigned

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

3 NFSSC Review

The NFSSC 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 NFSSC 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 NFSSC 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 Assessment Practices Manual will be updated to reflect the decisions. The effective date of the decisions is the date of the next January release of the manual.

Related IFPUG Documentation

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

The following table describes the other publications.

Document	Description
IFPUG Brochure (Available)	This publication is an introduction to the International Function Point Users Group. It includes a brief history of the organization, introduces function point analysis, and defines the purpose of IFPUG. The brochure also includes a membership application. Audience: This publication is for anyone who wants an overview of IFPUG or an application for membership.
IFPUG: Organizational Structure and Services (Available)	This publication describes IFPUG services, and lists the board of directors, committees, and affiliate members worldwide. Audience: This publication is for anyone who wants background information about IFPUG.
Counting Practices Manual (Release Date: January 2010)	International Function Point Users Group (IFPUG) Function Point Counting Practices Manual (CPM), Release 4.3.1
Guidelines to Software Measurement (Release Date: August 2004)	This manual provides an overview of software metrics for organizations working to create or improve software measurement programs. The manual addresses both system and customer management, provides high-level justifications for software measurement, and examines the components of effective measurement programs. Audience: This manual is intended for IFPUG members, Function Point Coordinators, persons who prepare the reports to management, and other persons knowledgeable about and working directly with function points.
Quick Reference Counting Guide v2.0 (Release Date: 2009)	This quick reference guide is a summary of function point counting rules and procedures. Audience: This summary information is intended for anyone applying function point analysis.
Adjusted Functional Size Quick Reference Guide (Release Date: 2010)	This quick reference guide is a summary of the General Systems Characteristics. Audience: This is intended for anyone using the optional General Systems Characteristics.
IFPUG Glossary (Available with CPM and Guidelines for Software Measurement)	This is a comprehensive glossary that defines terms used across IFPUG publications. Audience: The glossary is recommended for anyone who receives any of the other IFPUG documents or anyone who needs definitions of IFPUG terms.
A Framework for Functional Sizing, IFPUG, September 2003	This paper explains that product size contains three dimensions: functional size, technical size and quality size. The IFPUG FPA-method provides a measure for the functional size.

IT Measurement: Practical Advice from the Experts, Addison-Wesley, April 2002	This book is an excellent compilation of articles written by experts in the field of Information Technology. It was compiled by IFPUG to include recent insights in the application of software metrics in practice.
The IFPUG Guide to IT and Software Measurements (Release Date: 2012)	The IFPUG Guide to IT and Software Measurements brings together 52 leading software measurements experts from 13 different countries who share their insight and expertise. Covering measurement programs, function points in measurement, new technologies, and metric analysis, this volume: • Illustrates software measurement's role in new and emerging technologies • Addresses the impact of agile development on software measurements • Presents measurements as a powerful tool for auditing and accountability Include metrics for the CIO
Function Point Analysis Case Studies (Release Dates: Case Study 1 Release 3.0 September 2005 (CPM 4.2) Case Study 2 Release 3.0 March 2006 (CPM 4.2) Case Study 3 Release 2.0:September 2001 (CPM 4.1) Case Study 4 Release 2.0 September 2005 (CPM 4.2)	The case studies illustrate the major counting techniques that comprise the Function Point Counting Practices Manual. The cases illustrate function point counts for a sample application. The cases include the counting that occurs at the end of the analysis phase of software development and after system construction. Audience: The case studies are intended for persons new to function point analysis as well as those with intermediate and advanced experience.
Metric Views, Volume 6 Issue 2, August 2012	The Next Frontier: Measuring and Evaluating the Non-Functional Productivity
Metric Views, Volume 8 Issue 1, February 2014	Using SNAP for FUI Creation and Enhancement
Metric Views, Volume 8 Issue 1, February 2014	Experience of a SNAP user
APM 2.1 Quick Reference Guide - English (Release Date: 2014)	This quick reference guide is a summary of SNAP counting rules and procedures.
SNAP Case Study 1	A SNAP case study that focuses on sizing a requirement under sub-category 1.2 Logical and Mathematical Operations

Training Requirements

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

A workshop of two days has been developed to provide training in SNAP by IFPUG.

IFPUG has developed an official training certification and a training path for SNAP.

SNAP Certification

The CSP (Certified SNAP Practitioner) exam, provided by IFPUG, is a test of both the knowledge of the counting rules laid out in the APM and the ability to apply those rules. The exam consists of two sections: Definition and Implementation.

Part 1 –The SNAP Method

This page intentionally left blank

Introduction to SNAP

Introduction This chapter presents the assessment framework for non-functional sizing and a description of the objectives and benefits of SNAP.

Contents This chapter includes the following sections:

A Framework for Non-Functional Sizing

The main objective of IFPUG's Framework for Non-Functional Sizing (2008) project was to ensure that a non-functional framework can be used to establish a link between non-functional size and the effort to provide the non-functional requirements.

The resulting framework has the following characteristics:

- The overall framework is an assessment of the size of non-functional requirements
- The framework is comprised of assessment categories and sub-categories
- Sub-categories are evaluated using specified criteria
- The evaluation utilizes both assessed and/or measured criteria

The non-functional assessment results have the following characteristics:

- They can be used in conjunction with the functional size, and will help explain the variance in development effort and productivity
- Along with functional size, they can be used as input to estimating models
- They are determined from the users' non-functional view, but understood and agreed by the users and by the development teams.

Relationship between APM and Other Standards

IFPUG's Assessment Practices Manual (APM) for the Software Non-Functional Assessment Process (SNAP) uses definitions and terminology from relevant international standards organizations such as ISO, IEC and IEEE wherever possible.

A set of base definitions is given in the following section, regarding the classification of requirements.

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.)

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 9126-1:2001 defines functionality as follows:

Functionality is the set of attributes that bear on the existence of a set of

Functions and their specified properties. The functions are those that satisfy stated or implied needs.

ISO/IEC 14143-1 does not provide a definition for 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.

Examples: software performance requirements, software external interface requirements, software design constraints, and software quality attributes.

Note: Non-functional requirements are sometimes difficult to test, so they are usually evaluated subjectively.

Again and analogously, ISO/IEC/IEEE 24765 defines a Functional Requirement as:

- A statement that identifies what a product or process must accomplish to produce required behavior and/or results. [ISO/IEC 26710:2007 (IEEE std. 1220-2005) IEEE Standard for the Application and Management of the Systems Engineering Process. § 3.1.1.6.]
- A requirement that specifies a function that a system or system component must be able to perform. [ISO/IEC 24765, Systems and Software Engineering Vocabulary.]

ISO/IEC 25010:2011

The product quality model categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability).

Each characteristic is composed of a set of related sub-characteristics.

- **Functional Suitability**

  Degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.
  - Functional Completeness
  - Functional Correctness
  - Functional Appropriateness
- **Performance Efficiency**

  Performance relative to the amount of resources used under stated conditions.
  - Time behavior
  - Resource utilization
  - Capacity
- **Compatibility**

  Degree to which a product, system or component can exchange

information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.

- Co-existence
- Interoperability

- **Usability**

  Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

  - Appropriateness recognizability
  - Learnability
  - Operability
  - User error protection
  - User interface aesthetics
  - Accessibility

- **Reliability**

  Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

  - Maturity
  - Availability
  - Fault tolerance
  - Recoverability

- **Security**

  Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

  - Confidentiality
  - Integrity
  - Non-repudiation
  - Accountability
  - Authenticity

- **Maintainability**

  Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.

  - Modularity
  - Reusability
  - Analyzability
  - Modifiability
  - Testability

- **Portability**

  Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

  - Adaptability
  - Installability
  - Replaceability

Relationship between APM and ISO/IEC 9126-1, ISO/IEC 25010

SNAP sub-categories do not define or describe non-functional requirements; they classify how these requirements are met within the software product.

A nonfunctional requirement, which is defined either by ISO/IEC 9126-1 or ISO/IEC 25010, may be implemented in the product by using more than one sub-category. Accordingly, a sub-category may cater for several quality characteristics, as defined either by ISO/IEC 9126-1 or ISO/IEC 25010.

A view of the relationship is presented in the example and in table 1-1 below.

Example

This example analyses a requirement to improve the way system recovers from a crash:

Using ISO/IEC 25010, this requirement falls under the area of Reliability, and the attribute is Recoverability

SNAP sizes this requirement according to the design:

- An algorithm is added to identify corrupted data in specific fields.
- Time stamps are added to database records.
- An algorithm is written to reconstruct corrupted data using uncorrupted record.

The design involves the following SNAP sub-categories:

- Database Technology (adding time stamp).
- Logical and Mathematical operations.

In this example, the "recoverability" type of requirement is mapped to two sub-categories, "Database Technology" and "Logical and mathematical operations.

More examples of such mapping are illustrated in table 1-1.

Note:

All examples in table 1-1 assume that these requirements are not covered by Function Points.

To learn more about the link between FPA and SNAP, please refer to Appendix B and tables 3-1 and 3-2

		SNAP Categories and sub-categories														
		Data Operations					Interface Design				Technical Environment			Architecture	
ISO 25010 Sub-characteristics (A partial list for demonstration)		Data entry validations	Logical and Mathematical operations	Data Formatting	Internal Data Movements	Delivering Added value to users by Data	User Interfaces	Help Methods	Multiple Input Methods	Multiple Output Methods	Multiple platforms	Database Technology	Batch Processes	Component based software	Multiple Input/output Interfaces
		1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	3.1	3.2	3.3	4.1	4.2
Portability	Replaceability														
	Installability														
	Adaptability														
Security	Changeability														
	Authenticity	**Ex. 2**													
Reliability	Fault tolerance														
	Recoverability														
Usability	Accessibility														
	User error protection														
	Operability								**Ex. 4**						
	User interface aesthetics						**Ex. 1**								
	Learnability														
	Appropriateness recognizability														
Performance Efficiency	Capacity														
	Resource utilization														
	Time behavior											**Ex. 3**			
Functional Suitability	Functional Completeness														
	Functional Appropriateness														
	Functional Correctness														

Table 1-1 Mapping ISO/IEC 25010 characteristics to SNAP sub categories

The example numbers in Table 1-1 refer to the following:

Ex. 1. Improving the understandability and learnability by adding pop-up help menus (Sub-category 2.1) and rearranging the screens (Sub-category 2.2)

Ex. 2. Improving Security by adding more validations to authentication process, using Sub-category 1.1 “data entry validations” and Sub-category 1.2 “Logical and Mathematical operations”

Ex. 3. Improving performance by adding indices to the database and improving queries (Sub-category 3.2)

Ex. 4. Adding Barcode reading as additional input method (Sub-category 2.3)

Project Effort

Software Project

A collaborative enterprise, which is planned and executed to achieve a particular aim mainly by means of software development.

ISO/IEC/IEEE 16326:2009 "Systems and Software engineering – lifecycle processes – Project management" (Formerly IEEE Std-1058-1998, "IEEE Standard for Software Project Management Plans"): The set of work activities, both technical and managerial, required to satisfy the terms and conditions of a project agreement

Project effort is invested to provide the following:

- Features that are built into the product to meet the functional requirements (See definition in previous chapter).
- Features that are built into the product to meet the non-functional requirements (See definition in previous chapter).
- Project-related tasks, to ensure that
  - The project is managed.
  - The project meets its quality, timeline and budget constraints.
  - Risks are managed.

Meeting functional requirements (FUR) – the Product functional size

The functional size could be measured/assessed using the IFPUG functional size measurement method, based on the FURs.

Meeting non-functional requirements (NFR) –the Product non-functional size

The SNAP framework provides the basis for sizing non-functional requirements.

Performing project-related tasks

Project related tasks do not affect the Product size. Although these tasks affect the effort required to deliver the product, they influence the productivity and not the software size.

Examples of project related tasks:

- Team training.
- User training.
- Project documentation (such as manuals, plans, status reports, roadmaps, work instructions, quality standards)

Function Points Analysis (FPA) and SNAP result in different size-measures representing different dimensions of product size. While these sizes cannot be added together because they represent different dimensions (like volume and temperature of a room), they can be used together in estimating the effort towards the development of an application or a system.

In addition, project related tasks are also used to estimate the effort to develop the application or system.

SNAP Framework

The SNAP framework and FPA can be seen as three dimensions of a block:

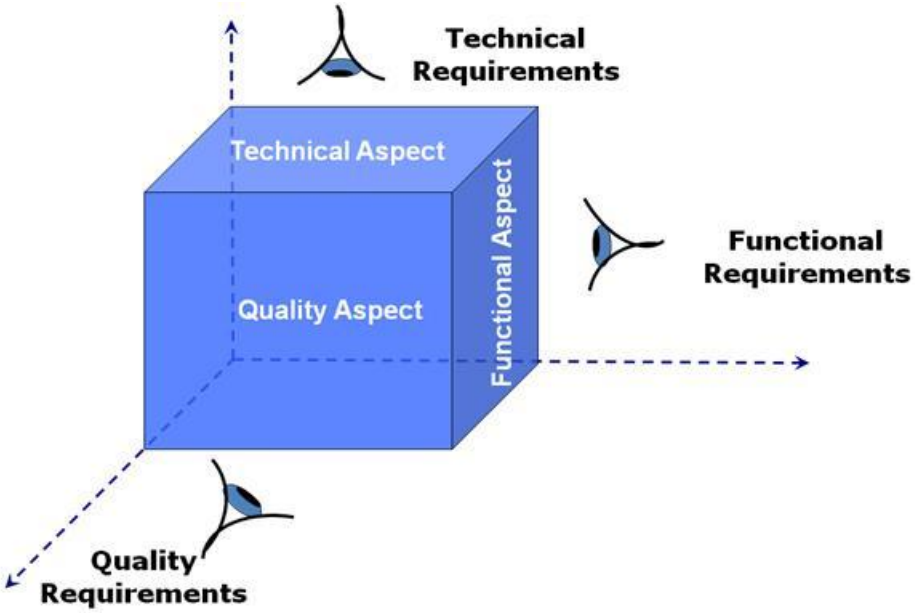


Figure 1-1 The Framework and Requirements

The dimension for the functional requirements (FUR) and the functional perspective of the software development project are currently covered by the functional size measure - IFPUG function points as defined in the IFPUG Counting Practice Manual. The dimensions of the non-functional requirements (both technical and quality) and the non-functional perspectives of the software development project are defined in this manual.

The following are the standards organizations definitions of technical requirements:

- ISO – requirements relating to the technology and environment, for the development, maintenance, support, and execution of the software.
- IEEE – combination of design, implementation, interface, performance, and physical requirements.

The Quality Requirements are requirements that are not defined as functional or technical, and relate to the quality of the system or components. The following are available definitions:

- ISO – The following characteristics form part of the quality model: Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability.
- IFPUG – Quality includes: conformity to user expectations, user requirements, customer satisfaction, reliability, and level of defects present.

SNAP Objectives and Benefits

Objectives

SNAP measures software by quantifying the size of non-functional requirements. With this in mind, the objectives of SNAP are to:

- Measure the non-functional size of the software that the user requests and receives
- Demonstrate the full economic value of the application, including its functional aspects as well as the non-functional aspects (have the non-functional baseline as well as the functional baseline, to demonstrate the full economic value)
- Measure software development and maintenance based on the non-functional requirements (such as technology used for implementation)
- Size technical projects, in which FPA is not applicable

In addition to meeting the above objectives, the process of assessing non-functional requirements should be:

- Simple enough to minimize the overhead of the measurement process.
- A consistent measure among various projects and organizations. SNAP allows to determine (by counting each of the four categories, from each one of the sub-characteristics) the possibility to size and therefore better estimate a project with/without FPs, according to the set of user requirements received for a project.

Benefits

A non-functional assessment will assist IT organizations in multiple ways. It will provide insight into the delivery of projects and maintenance of applications to assist in estimating and in the analysis of quality and productivity. Used in conjunction with FP measures, the non-functional assessment will provide information that can identify items impacting quality and productivity in a positive or negative way.

Having this information enables software professionals to:

- Better plan and estimate projects.
- Identify areas of process improvement.
- Assist in determining future non-functional strategies.
- Quantify the impacts of the current non-functional strategies.
- Provide specific data when communicating non-functional issues to various audiences.

Organizations can apply SNAP as:

- A methodology to measure the non-functional size of a software product to support quality and productivity analysis.
- A methodology to estimate cost and resources required for software development and maintenance.
- A methodology to measure cost reduction for software development and maintenance, in addition to FPA.

- A normalization factor for software comparison.
- A methodology to determine the non-functional size of a purchased application package by assessing all the portions and categories included in the package.
- A methodology to help users determine the benefit of an application package to their organization by assessing portions or categories that specifically match their requirements.

Note: "FP + SNAP points" are **not** equal to the overall product size.

As of the date of this publication, the size of a software application is considered to have two distinct parts: the size of the FURs and the size of the NFRs. For example, if an application's functional size is 700 function points and non-functional size is 200 SNAP points, then the entire size could be stated as "700 function points, and 200 SNAP points." The two sizes do not sum up to one single size such as "900 points."

The IFPUG functional sizing methodology does not change when measuring the non-functional requirements using SNAP.

A project may have 0 FP and non-zero number of SNAP points, or 0 SNAP Points and non-zero number of FP, or any combination of FP and SP

Further research is needed to determine if function points and SNAP points can be somehow combined as part of single metric.

This page intentionally left blank

SNAP Overview

Introduction This chapter presents an overview of SNAP. It presents a summary of the related concepts and process steps.

Contents This chapter includes the following:

SNAP Description

Introduction

The purpose of this chapter is to describe how non-functional requirements can be sized using SNAP method.

Looking at the various scenarios on non-functional aspects within an application, various sub-categories are identified and are grouped under logical categories.

The categories and sub-categories do not replace or explain the standards that describe and classify the non-functional requirements (such as ISO/IEC 25010:2011). The categories and sub-categories describe how the assessed project or product will meet these non-functional requirements.

Non-Functional Assessment Process

The non-functional assessment will use a series of questions grouped by categories to measure the size of non-functional requirements for the development and delivery of the software product.

- The categories will focus on those non-functional requirements that affect product size.
- The process will allow for the sizing of the non-functional requirements using a series of questions and measures.
- The process can be used for Development Projects, Enhancement Projects, Maintenance Activities, and Applications.

Procedure by Section

The following table shows the SNAP procedure as explained in the remaining chapters of Part 1.

Procedure	Section
Determine Assessment Purpose, Scope, Boundary and Partition	1
Associate non-functional requirements with categories and sub-categories	2
Identify the SNAP Counting Units (SCUs)	3
Determine the complexity of Each SNAP Counting Unit (SCU)	4
Determine the SNAP Points (SP) of each SCU	5
Calculate Non-Functional Size (SNAP Points)	6

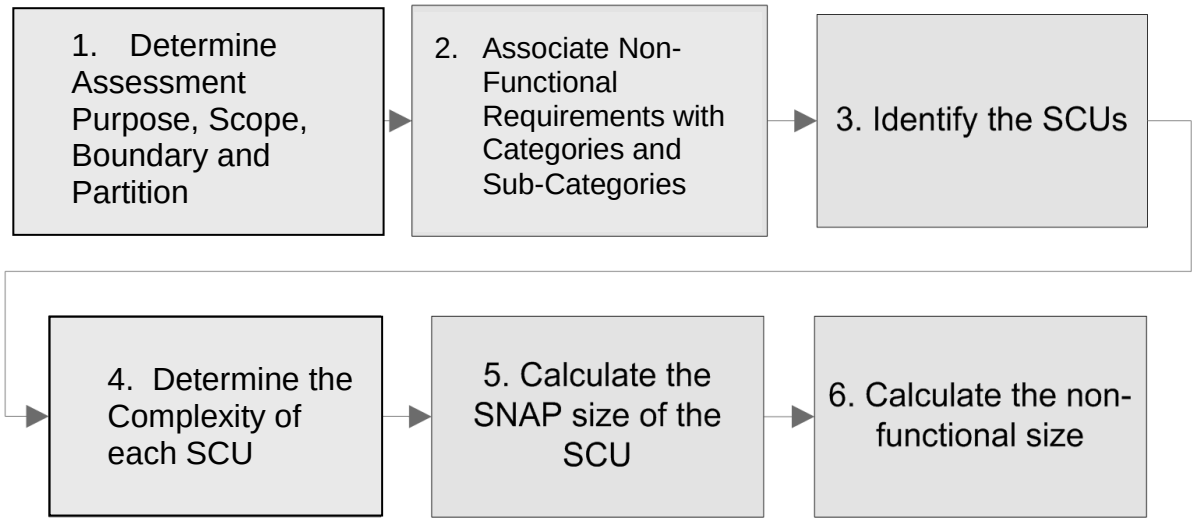


Figure 1-2 SNAP Diagram

Section 1: Determine Assessment Purpose, Scope, Boundary and Partition

The following steps shall be performed when identifying the Assessment Purpose, Scope, Boundary and Partition:

Step	Action
1	Identify the purpose of the assessment
2	Identify the assessment type
3	Identify the assessment scope
4	Identify the boundary of the application(s)
5	Identify the partitions, if applicable
6	Document purpose, type, scope, boundary, partition, and assumptions

See Chapter 4 for more details about the purpose, scope, and boundary of the assessment.

This section defines the assessment type, assessment scope, boundaries and partition

Assessment Type The functional size and the non-functional size can be measured for either projects or applications. The type of assessment is determined, based on the purpose, as one of the following:

- Development project assessment.
- Enhancement project assessment.
- Application assessment.

Chapter 4 in Part 1 defines each type of assessment.

Boundary 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 pass into and out of the application.
- Is dependent on the user's external business view of the application; it is independent of non-functional and/or implementation considerations.

Partition

A partition is a set of software functions within an application boundary that share homogeneous assessment criteria and values. A partition requires development effort, that may not be reflected when sizing the functional aspect of the project/product, using FPA

The positioning of the partition may be subjective. It is often difficult to delineate where one partition stops and another begins. Try to place the partition from a non-functional perspective of the users, such as maintainability, portability, or installability, rather than based on technical or physical considerations. It is important that the partition is placed with care, since all data crossing the partition impact SNAP size.

- The partition is determined based on the user view. The focus is on what the user can understand and describe.

Within a boundary, partitions:

- Contain all the software functions which constitute the overall functionality of the application being assessed.
- May cooperate between themselves in order to provide complete software functions to the application user.
- Shall not overlap.
- Shall be consistent over time

A partition:

- May be used to meet non-functional requirements.
- Can be sized using SNAP categories and sub-categories.
- Might coincide with the overall application (such as: client and server residing in a single system).

In case where there are no identifiable partitions the boundary itself is taken, and no partitions are considered.

Examples of partitions

Examples of partitions are:

- The client functions in a client-server application
- The server functions in a client-server application.
- The functions of "user A", (to be) separately designed and/or implemented from functions of "user B", within the same application.
- The functions (to be) implemented over non-functional platform "X", separately identified from functions (to be) implemented over non-

functional platform "Y", within the same application.

- SOA – Application within the boundary.
- Peer component within the boundary.

General Note: Counting SNAP points is performed at boundary level. Partitions add SNAP size that refer to the internal data movements (See Figure 1-3)

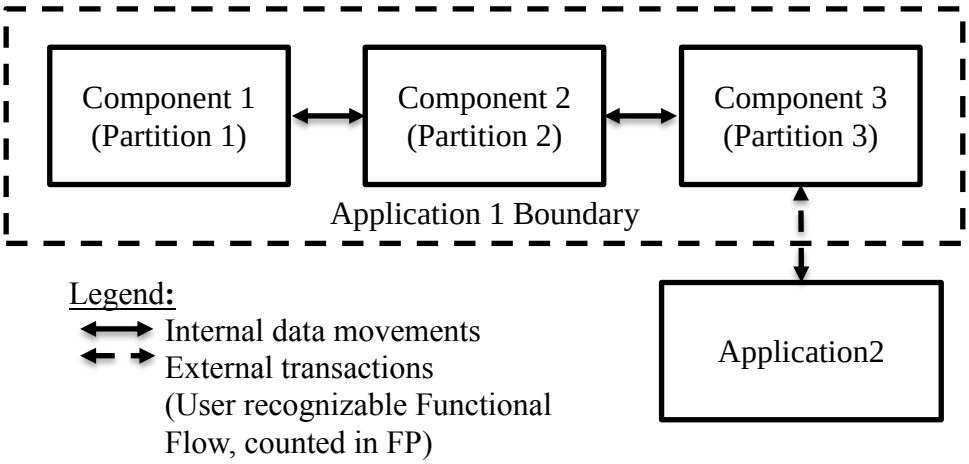


Figure 1-3 Relations between partitions and applications

Section 2: Associate Non-functional Requirements with Categories and Sub-categories

This section defines the categories and sub-categories, and describes the association of the non-functional requirements with sub-categories. The categories and sub-categories are standard for any SNAP assessments.

Category Definition A category is a group of components, processes or activities that are used in order to meet the non-functional requirement.

- Categories classify the non-functional requirements.
- Categories are generic enough to allow for future technologies.
- Categories are divided into sub-categories. Each sub-category has common features (within the sub-category). This simplifies the assessment.

Each SNAP category groups the sub-categories based on the same level of operations and/or similar type of activities executed by the non-functional assessment process.

Sub-category Definition

Sub-category is defined as a component, a process or an activity executed within the project, to meet the non-functional requirement.

Note: A non-functional process may have to execute more than one sub-category to meet the non-functional requirement.

Category & Sub-category

The association of the non-functional requirements with categories and sub-categories is performed as follows:

- Identify the non-functional requirement under scope (for example, requirements for data security; requirements to improve performance)
- Analyze the design and identify which sub-categories are used in order to meet the requirement

Categories and sub-categories are:

1. Data Operations
   - 1.1. Data Entry Validations.
   - 1.2. Logical and Mathematical Operations.
   - 1.3. Data Formatting.
   - 1.4. Internal Data Movements.
   - 1.5. Delivering Added Value to Users by Data Configuration.
2. Interface Design
   - 2.1. User Interfaces.
   - 2.2. Help Methods.
   - 2.3. Multiple Input Methods.
   - 2.4. Multiple Output Methods.
3. Technical Environment
   - 3.1. Multiple Platforms.
   - 3.2. Database Technology.
   - 3.3. Batch Processes.
4. Architecture
   - 4.1. Component Based Software.
   - 4.2. Multiple Input / Output Interfaces.

The sub-categories address the non-functional requirements, including Technical and Quality requirements. (Quality requirements such as usability or reliability, as defined by ISO/IEC 9126-1 or ISO/IEC 25010, can be addressed by the following sub-categories: Data Entry Validation, Data Formatting, and User Interface. For example, adding On-line Help is a requirement to improve ease of learning and can be sized by using the sub-category "Help Methods.")

See Chapter 5 for more details about the categories and sub-categories.

Section 3: Identify the SNAP Counting Units (SCUs)

This section defines the rules and procedures that apply when identifying the SNAP Counting Units (SCUs).

The SCUs are unique to each sub-category; they are determined by the nature of the sub-category.

The SCU is part of the sub-category definition.

Sizing is done separately per each SCU.

Note: A requirement may contain both functional and non-functional aspects. In such a case, the requirement will have a functional size, measured in Function Points, and a SNAP size, measured in SNAP points.

Use Part 3, Appendix B ("Counting Function Points and SNAP points") for requirements that involve both functional and non-functional requirements.

The example in Part 2 demonstrates how SCUs are used.

SNAP Counting Unit (SCU) Definition

The SCU is a component or activity, in which complexity and size is assessed.

The SCU can be a component, a process or an activity identified according to the nature of the sub-category/sub-categories.

An SCU may contain both functional and non-functional characteristics. In these cases, sizing of the elementary process will be performed for its functional sizing, using function point analysis, and for its non-functional sizing, using SNAP.

See Chapter 5 for more details about the SCU.

Section 4: Determine the Complexity of each SNAP Counting Unit (SCU)

This section defines how to determine the complexity and size of each SCU within the sub-category.

Answer the assessment questions for each sub-category.

The assessment questions are related to parameters that affect the complexity of a given sub-category.

The assessment rating is the answer to the assessment questions.

The complexity level of an assessment rating or the value of the parameters within each SCU is mapped to a size.

See Chapter 5 for more details about the determination of complexity.

Section 5: Determine the SNAP Points (SP) of Each SCU

This section defines how to determine the size of each sub-category.

SNAP Points Definition

SNAP points (SP) are the sum of the size of all SCUs identified in each sub-category.

Once all complexity parameters have been assessed, the size of each SCU is calculated and the SP of all SCUs are added together to obtain the calculated SP for the sub-category.

See Chapter 5 for more details how to calculate the SP for each SCU.

Section 6: Calculate Non-functional Size

This section defines how to calculate the size of the non-functional aspect of the project/product in scope.

The SPs are the final non-functional size obtained by combining all category values.

When more than one category is identified, the overall non-functional size (SP) shall be obtained by combining the size of each sub-category within the application boundary of the software product being assessed.

Note:

When many (more than one) sub-categories are using the same definition of SCU, such as the elementary process, then we should answer the assessment questions from all relevant sub-categories for this elementary process.

Counting will be per SCU per sub-category. Therefore an SCU may generate SP from more than one sub-category. Requirement # 1 in the case study that is presented as an example.

See Chapter 6 for more details about the calculations and the definition of the formulas.

Assessment Preparation

Introduction This chapter presents the concept of the user's role in defining the non-functional requirements for a project or application. This chapter also includes a description of useful documentation during the life cycle of an application.

Contents This chapter includes the following:

Timing of Non-functional Assessments

Non-functional assessments can be completed at any time in the development life cycle to aid in project estimating, monitoring project change of scope, and evaluating delivered non-functional requirements.

Prior to beginning a non-functional assessment, determine whether you are approximating or measuring the size and document any assumptions.

Approximating permits assumptions to be made about unknown non-functional categories and/or their complexity in order to determine an approximate non-functional size.

Measuring includes the identification of all applicable non-functional sub-categories and their complexity to accomplish a non-functional size analysis.

At an early stage non-functional requirements may not be fully defined. Despite the disadvantages, this assessment can be very useful to produce an early estimate. Uses of the non-functional assessment for approximating or measuring non-functional size at the various life cycle phases are presented below:

Life Cycle Phase	**SP can be approximated**	**SP can be measured**
Proposal: users express needs and intentions	Yes	No
Requirements: developers and users review and agree upon expression of user needs and intentions	Yes	No
Design: developers may include elements for implementation	Yes	Yes
Construction	Yes	Yes
Delivery	Yes	Yes
Maintenance (Adaptive – modifying the system to cope with changes in the software environment. Perfective – implementing new or changed user requirements which concern functional enhancements to the software. Preventive – increasing software maintainability or reliability to prevent problems in the future)	Yes	Yes
Maintenance (Corrective - Reactive modification of a software product performed after delivery to correct discovered problems)	SP not used	SP not used
Note: No specific development life cycle is implied. If using an iterative approach, you may expect to approximate/measure the SP multiple times during the project life cycle.		

Table 1-2 Timing of SNAP Assessment

Useful Project/Application Documentation

Typically software development life cycle activities include the development of a Technical Requirements document and/or a Non Functional Requirements document. The Technical Requirements may include elements which are necessary for the implementation, but which are not considered in functional size measurement (e.g. temporary files, index, etc.).

This document may have one or more of the following characteristics:

- Technology dependence.

  For example, physical files vary based on the database environment.
- Terminology unfamiliar to the users

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

  For example, the computing capacity (infrastructural support to aid processing) currently available in the organization.
- Physical boundaries.

  For example, there may be separate technical requirements for client and server

In general, the following items are useful when conducting any non-functional assessment:

- Requirements documents.
- Entity relationship diagrams.
- Technical requirements documents
- Object models.
- Physical data models
- Physical file and database layouts.
- Interface agreements with descriptions of batch feeds/transaction files and interfaces to/from other applications.
- Samples of reports, displays, and other user interfaces.
- Demonstration of application operation.
- UI standards.
- Availability of one or more technical experts will help in a great way (for the application being assessed).
- System design documentation.
- Technical design document.
- Architecture diagrams.
- Use cases/functional requirement document.

Note: The list above is not all-inclusive.

Estimated and Final Non-functional Assessment

It is important to realize that early non-functional assessments are estimates of to-be-delivered non-functional requirements. In addition, as the scope is clarified and the requirements developed / evolved, it is quite normal to identify additional non-functional characteristics, which were not specified in the original requirements. This phenomenon is sometimes called "scope creep." It is essential to update the application size upon completion of the project. If the size changes during development, then the non-functional size at the end of the life cycle should accurately reflect the full non-functional characteristics delivered to the user.

Determine Purpose, Scope, Boundary and Partition

Introduction This chapter describes in detail the steps in the determination of scope and boundary. It explains how they are influenced by the purpose of the assessment and the type of assessment.

Contents This chapter includes the following sections:

Steps for Determination of Scope and Boundary

When identifying Scope and Boundary, the following steps shall be performed:

1. Identify the purpose of the assessment.
2. Identify assessment type, based on the purpose.
3. Determine the assessment scope, based on the purpose and type of count.
4. Determine the boundary of the application(s).
5. Determine the partitions, if applicable.
6. Document the following items:
   - The purpose and type of the assessment.
   - The assessment scope.
   - The boundary of the application(s).
   - The partition(s) within the boundary(s).
   - Any assumptions related to the above.

Consistency with FPA

The purpose, scope, and logical application boundaries need to be consistent between the FPA and SNAP process.

See Appendix B for the link between FPA and SNAP.

Identify the Purpose of the Assessment

A non-functional size measurement is conducted to measure the size of non-functional requirements for the development and delivery of a software product. In order to provide answers relevant to the purpose for the assessment, the assessment scope, boundaries, and partitions must be separately identified.

The purpose:

- Determines the type of non-functional assessment and the scope of the required assessment to obtain the non-functional size.
- Determine the size of the non-functional aspect of the assessed products.

Examples of purposes are:

- To provide the non-functional size of a development project as an input to the estimation process to determine the effort to develop the first release of an application.
- To provide the non-functional size of the installed base of applications to determine the support costs.
- To provide the non-functional size delivered by an enhancement.
- To provide the non-functional size for maintenance activities.

Identify the Type of Assessments

The functional size and the non-functional size can be measured for either projects or applications. The type of assessment is determined, based on the purpose, as one of the following:

- Development project assessment.
- Enhancement project assessment.
- Application assessment.

The following paragraphs define each type of assessment.

Development Project

A **development project** is a project to develop and deliver the first release of a software application.

DSP – Development Project SNAP points.

The **development project non-functional size** is an assessment of the non-functional requirements provided to the users with the first release of the software, as measured by the development project SNAP assessment by the activity of applying the SNAP method.

Enhancement Project

An **enhancement project** is a project to develop and deliver corrective, preventive, adaptive or perfective maintenance.

ESP – Enhancement Project SNAP points.

The **enhancement project non-functional** size is a measure of the non-functional characteristics added, changed or deleted at the completion of an enhancement project, as measured by the enhancement project SNAP assessment.

Application

An **application** is a cohesive collection of automated procedures and data supporting a business objective; it consists of one or more components, modules, or sub-systems.

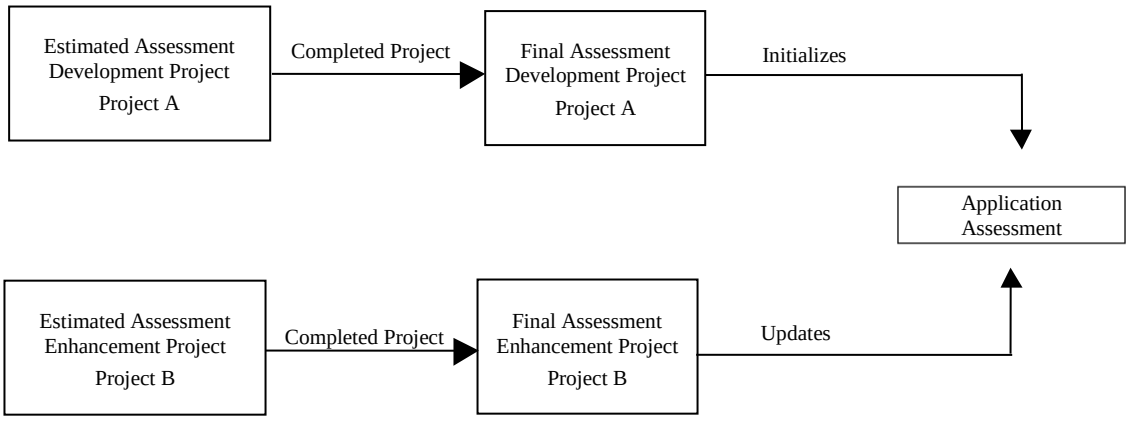
ASPA – The application SP after the enhancement project.

An **application's non-functional size** is a measure of the non-functional characteristics that an application provides to the user, determined by conducting the application SNAP assessment.

It is also referred to as the baseline or installed non-functional size. This size provides a measure of the current non-functional characteristics the application provides the user. This number is initialized when the development project SNAP assessment is completed. It is updated every time a completion of an enhancement project alters the application's non-functional size.

Diagram of Types of Assessments

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



The previous example shows the non-functional assessment concept but does not reflect the full example of the process.

Identify the Scope of the Assessment

The assessment scope defines the set of non-functional user requirements to be included in the assessment. The scope:

- Is determined by the purpose for performing the non-functional assessment.
- Defines a set of partition(s).
- Identifies which non-functional assessment categories and sub-categories will be included in the non-functional size measurement to measure the size of non-functional requirements for the development and delivery of the software product.
- Could include more than one application.

The scope of:

- A development project non-functional assessment includes all non-functional requirements for the development and delivery of the software product.
- An assessment of an installed base of applications includes all non-functional requirements for the support of the installed applications.
- An enhancement non-functional assessment includes all non-functional requirements for the development and delivery of the enhancement project; the boundary of the application(s) impacted remains the same.
- A maintenance assessment includes all non-functional requirements for a selected scope.

Determine the Boundary

Determine the boundary of each application within the assessment scope, based on the user view.

User view definition

In order to establish the boundary, the user view must be defined. The following hints can help you to identify the user view:

- A user is any person or thing (application, device, etc.) that communicates or interacts with the software at any time.
- A user view consists of the functional and non-functional requirements as perceived by the user.
- A partition may act as an "internal user" for another partition within the same application boundary, in terms of data exchange or data sharing; consequently, different non-functional assessments might be created for each partition.

A user view:

- Is a description of the business functions and non-functional requirements.
- Represents a formal description of the user's needs in the user's language.
- Can be verbal statements made by the user as to what their view is.
- Is approved by the user.
- Can be used to measure the functional and non-functional size.
- Can vary in physical form (e.g., catalog of transactions, proposals, requirements document, external specifications, detailed specifications, user handbook, quality or non-functional specifications).

Boundary definition

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 pass into and out of the application.
- Is dependent on the user's external business view of the application; it is independent of non-functional 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. Try to place the boundary from a business perspective rather than based on technical or physical considerations.

For example, the following diagram shows boundaries between the Human Resources application and the external applications, Currency and Fixed Assets. The example also shows the boundary between the human user (User 1) and the Human Resources application. The Human Resource application may in turn internally satisfy the functional, technical and quality requirements specified by the user.

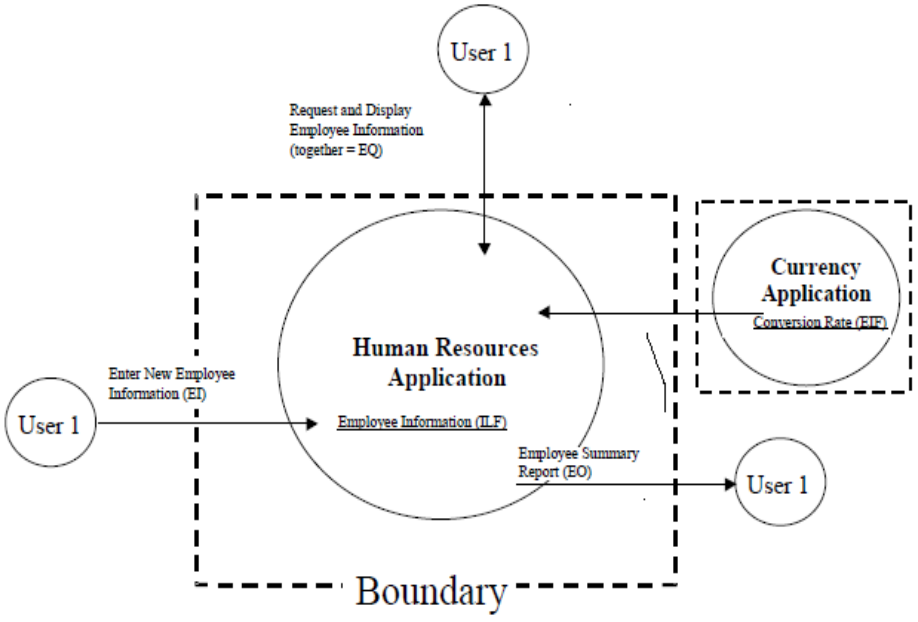


Figure 1-4 – Boundary example

Determine the Partition

Partition When identified, partitions may add non-functional size. Sub-category 1.4 (Internal Data Movements) is used to provide the additional non-functional size for the application being assessed.

Rules and Procedures

This section defines the rules and procedures that apply when determining assessment scope and boundary of the application(s).

Scope hints The following hints can help you to identify the assessment scope:

- Review the purpose of the non-functional assessment to help determine the assessment scope.
- When identifying the scope for the assessment of the non-functional size of the installed base of applications, include all of the non-functional categories supported by the maintenance team, eventually distinguished by partition within each application's boundary.

Boundary Rules

The following rules shall apply for boundaries:

- The logical application boundaries need to be consistent between the FPA and SNAP processes.
- The boundary is determined based on the user view; the focus is on what the user can understand and describe.
- The initial boundary already established for the application, or applications being modified, is not influenced by the assessment scope.

Note:

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

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

Boundary and Partitions Hints

The following hints can help you to identify the boundary and the partition of the application(s):

- Use the system external specifications or a system flow chart and draw a boundary around it to highlight which parts are internal and which are external to the application.
- Look at how groups of data and software partitions are being maintained.
- Identify functional areas by assigning ownership of certain types of analysis objects (such as entities or elementary processes) to a functional area; non-functional categories are determined by the identification of the functional boundaries (Application boundaries as determined by FPA) and eventually partitions within them.
- Look at the associated measurement data, such as effort, cost, and defects; the boundaries for measurement should be the same, or eventually those measurement data might be distinguished by partitions within a single boundary.
- Interview subject matter experts for assistance in identifying the boundary.
- Interview software analysts for assistance in identifying the partitions, if any.

This page intentionally left blank

Categories & Sub-categories

Introduction This chapter presents the details behind categories & sub-categories including questions, ratings, and SCUs.

Contents This chapter includes the following:

Categories and Sub-categories

Definition

See Chapter 2 Section 2 for the definition of categories and sub-categories

Sub-category Complexity

Complexity definition

Each sub-category is of a different nature; therefore, it may have a different set of parameters that define complexity. Complexity was defined by asking the following questions:

1. What are the main drivers that are considered by software project estimators as affecting the complexity of the item.

2. Assuming one small team has its one set of productivity values (skill set, methodologies, working environment etc.) – such a team will estimate that more work is needed to provide a complex item than to provide a medium item.

Complexity Parameters

The Parameters that are counted or evaluated in order to assess the Complexity.

In the example below (see table 1-3), the complexity parameters are:

- Number of Data Elements Types (DETs).
- Number of nesting levels.

Complexity Example

Complexity of Data Entry Validation may be defined by the number of nesting levels, and also by the amount of data elements that are used in the process. In this case, we may decide that parameter #1 will be based on the number of nesting levels, and parameter #2 will be the number of data elements.

Complexity grid

Use the following tables to illustrate SP calculation

	Nesting Level Complexity		
	Low	**Average**	**High**
	1-5	**6-14**	**15+**
SP=	2*#DETs	3*#DETs	4*#DETs

Table 1-3 Example 1 of SP calculation

Another option to illustrate SP calculations is based on the type of the assessed item, for example:

Help Type	SP =
User Manual	1*(#help items)
On-line Text	2*(#help items)
Context Help	2*(#help items)
Context + On-line	3*(#help items)

Table 1-4 Example 2 of SP calculation

Code Data

Code data is a type of data entities, used for software sizing (in addition to Business Data and Reference Data).

According to IFPUG CPM (Part 3 Chapter 1), "Code Data" usually exists to satisfy non-functional user requirements from the user (for quality requirements, physical implementation and/or a technical reason).

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 non-functional user requirements.

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' attributes describing the code; e.g., standard abbreviation, effective date, termination date, audit trail data, etc. The different categories of data are outlined below to assist in identification.

When codes are used in the Business Data, it is necessary to have a means of translating to convert the code into something more recognizable to the user. In order to satisfy non-functional user requirements, developers often create one or more tables containing the Code Data. Logically, the code and its related description have the same meaning. Without a description the code may not always be clearly understood.

The key difference between Code Data and Reference Data is:

- With Code Data, you can substitute one for the other without changing the meaning of the Business Data; e.g., Airport-Code versus Airport-Name, Color-Id versus Color-Description.
- With Reference Data, you cannot substitute (e.g., Tax Code with the Tax-Rate).

Code Data has most of the following characteristics:

Logical Logical Characteristics include:

- Data is mandatory to the functional area but optionally stored as a data file.
- Not usually identified as part of the functional user requirements; it is usually identified as part of design to meet non-functional user requirements.
- Sometimes user maintainable (usually by a user support person)
- Stores data to standardize and facilitate business activities and business transactions.
- Essentially static - only changes in response to changes in the way that the business operates.
- Business transactions access Code Data to improve ease of data entry, improve data consistency, ensure data integrity, etc.
- If recognized by the user:
    - Is sometimes considered as a group of the same type of data.
    - Could be maintained using the same processing logic.

Physical Physical characteristics include:

- Consists of key field and usually one or two attributes only.
- Typically has a stable number of records.
- Can represent 50% of all entities in Third Normal Form.
- Sometimes de-normalized and placed in one physical table with other Code Data.
- May be implemented in different ways (e.g., via separate application, data dictionary, or hard-coded within the software).

Examples Examples of Code data include:

- State.
    - State Code.
    - State Name.
- Payment Type.
    - Payment Type Code.
    - Payment Description.

Handling Code Data from Non-functional Sizing Perspective

For the purpose of FPA, Code Data cannot be counted as logical files. They are not considered as Internal Logical File (ILF) or External Interface File (EIF), and cannot be considered Record Element Types (RETs) or Data Element Types (DETs) on an ILF or EIF. Code Data cannot be considered a File Types Referenced (FTR) while assessing the complexity of a transactional function (External Input -EI, External Output - EO, and External Inquiry - EQ).

For the purpose of SNAP, Code Data which is maintained within the application boundary by use of screens or by formal enhancement requests by the customer is counted as follows:

- Irrespective of the number of Code Data physical tables, the Code Data would be

grouped as 1 data group (1 FTR) under SNAP.

- Code data is classified as

Types of Code Data

Substitution	Static or Constant	Valid Values
Code + Description	One occurrence Static Data Default value	Valid Values Range of valid values

Table 1-5 Types of Code Data

- For SNAP analysis of complexity of Code Data group, the number of RETs of the code table would depend upon the type of occurrences of Code Data types.

Example

In a Banking application, the following Code tables were created:

1. Table 1: Having State name and State code
2. Table 2: Having Branch code, Branch name, Branch city
3. Table 3: Having one single data entry of the Bank Name and logo , which is used for printing letterheads

Three types of occurrences exist for Code Data

1. Substitution - Table 1 having State code and state name, Table 2 having branch code and branch name
2. Valid Values -Table 2 having range of bank branch cities
3. Static Data - Table 3 having one single data entry of the Bank Name and logo , which is used for printing letterheads

Hence the number of RETs for the Code data group is 3.

Note: If the data is not of the above code data sub types, then it may be the data in system tables and not the application tables required for supporting business. This is not sized under code data

How Code Data is sized using SNAP?

Count the creation / maintenance and the utilization of code data:

- The creation of Code Data is always counted as "3.2 Database Technology."
- The maintenance of Code Data is checked under the following subcategories depending on the case applicable
    - If the Code Data is maintained in hard coded tables which are not viewable via screens but can be updated by the administrator using script/code change in response to a formal user request, then it is counted using sub-category 3.2: Database technology.
      For example, a code table has the bank name and logo stored as static data, which is referred by different processes. When a change request is raised to modify the logo, then it is sized using this category.
    - When the Code Data is used for reasons such as entry validations, data formatting, batch process management, any changes to code data values (Add /Change/ Delete) will be counted using the proper sub-category.

- The utilization of Code Data is counted in the following sub-categories, according to the purpose of the data: "1.1 Data Entry Validation"; "1.2 Logical and Mathematical Operations"; "1.3 Data Formatting"; "1.5 Delivering Added Value to Users by Data Configuration" and "3.3 Batch Processes."
- When non-functional requirements use Code Data and transactions cross partitions, sub-category "1.4 Internal Data Movements" should be used.

Examples of SNAP sizing of Code Data

Examples:	Sub-Categories for Utilizing Code Data
1. Create a Code Data table for address validation	1.1 Data Entry Validation for enabling the validation 3.2 Database Technology for code table creation
2. Same as above. The screens with the address are on Front-End application, the data is in the Back-End application	1.1 Data Entry Validation for enabling the validation 3.2 Database Technology for code table creation 1.4 Internal Data Movements
3. Using multi-language screens, the translation is in new Code Data tables	1.3 Data Formatting 3.2 Database Technology for code table creation
4. Add scheduling data to perform batch files	3.3 Batch Processes 3.2 Database Technology for code table creation

Table 1-6 Example of SNAP sizing of Code Data

Refer to the subcategories and to the examples in Part 1, Chapter 5 below

Definitions of Additional Terms used in this Manual

Elementary Process (EP) An elementary process is the smallest unit of activity that is meaningful to the user(s). The elementary process must be self-contained and leave the business of the application being counted in a consistent state

Data Element Type (DET) **A DET** (Data Element Type) in this manual, is a unique, non-repeated attribute, which can be in Business Data, Reference Data, or Code Data.
Count the number of different types of Data Elements of all tables as the number of DETs.

# of DETs The sum of all DETs which are part of the input + output of the elementary process, plus the data elements which are read or updated internal to the boundary.

Record Element Type (RET)

User recognizable sub-group of data element types within a data function, and Code Data group as defined in the "Code Data" paragraph.

Logical File

A logical file is a logical group of data as seen by the user. A logical file is made up of one or more data entities.

A data function represents functionality provided to the user to meet internal and external data storage requirements. A data function is either an internal logical file or an external interface file.

Grouping of data into logical files is the result of the combined effect of two grouping methods:

- Method a) is process driven, based on the user transactions in the application.
- Method b) is data driven, based on the business rules.

File Type Referenced (FTR)

A file type referenced is a data function read and/or maintained by a transactional function. A file type referenced includes:

- An internal logical file read or maintained by a transactional function, or
- An external interface file read by a transactional function
- Code Data is grouped to one additional FTR

Database View

In database theory, a database view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary. Unlike ordinary base tables in a relational database, it is a virtual table computed or collated dynamically from data in the database, when access to that view is requested. Changes applied to the data in a relevant underlying table are reflected in the data shown in subsequent invocations of the view.

Constant Factor

A multiplier used to calculate the number of SNAP points. The SNAP point size is a result of the [constant factor] times [a complexity parameter].

Example:

SP =	2*#DETs

2 is the constant factor

#of DETs is the complexity parameter

Single / Multiple Instance approach

Different organizations may take different approaches for sizing similar functionality being delivered on different medias. They use the single instance or the multiple instance approaches to specify the same. Single instance approach is said to be when the same functionality is delivered via different mediums (input or output), but is counted only once.

Multiple instance approach is the case where each method of delivery of same functionality is counted separately.

Organizations using the single instance approach for the FP size can size the other methods of delivery using SNAP.

Single Instance approach

The single instance approach does not recognize the medium for delivery for a transaction function as a differentiating characteristic in the identification of unique transaction functions. If two functions deliver the same functionality using different media, they are considered to be the same function for functional sizing purposes.

Multiple Instance approach

The multiple instance approach specifies that instance functional size is taken in context of the approach objective of the count, allowing a business function to be recognized in the context of the medium in which it is required to operate.

The multiple instance approach recognizes the medium for delivery for a transaction function as a differentiating characteristic in the identification of unique transaction functions.

Category 1: Data Operations

Data Operations The Data Operations category relates to how data is processed within the SCU to meet the non-functional requirements in the application.

1.1 Data Entry Validation

Definition Operations that are performed either to allow only certified (predefined) data or to prevent the acceptance of uncertified data

SCU The elementary process.

Terms

Nesting Level:

The number of conditional validations (If-Else combo/"While" loop/"For" loop or any other validation blocks) in the longest chain of validation

Complexity Parameters:

1. Nesting level complexity
   a. Low complexity: 2 nesting levels or less
   b. Average complexity: more than 3 nesting levels and less than or equal to 5
   c. High complexity: more than 6 nesting levels
2. Number of DETs used for validation. (The maximum number of DETs and not the sum of the DETs passing the different nesting levels)

SP calculation Identify the complexity based on nesting level. Calculate SP based on the constant factor and the number of DETs (#DETs).

	Nesting Level Complexity		
	Low	**Average**	**High**
	1-2	**3-5**	**6+**
SP =	2*#DETs	3*#DETs	4*#DETs

Table 1-7 SNAP sizing for Data Entry Validations

Notes:

- Data entry may result from any source (UI, transaction).
- Number of nesting levels is derived from the business requirements and high level solution and not from how the code is written.
- Validations are nested when there is a dependency between validations.
  Example: A number must be between 0 and 10: if it is less than one, it must have two digits after the decimal point. If it is one or more, it

may have one or no digits after the decimal point.

- Several validation on a field are not nested when they are independent.
  Example: a value must be numerical, greater than 0 and smaller than 1,000,000.
- This sub-category may include requirements for error handling or exceptions handling.
- DETs refer to all types of data. See Part 1 Chapter 5 (Definitions of Additional Terms) for the definition of data elements.
- If Code Data is used for data entry validations, then any changes to Code Data values (Add/Change/Delete) will be counted using this category.

Examples

1: Data Entry validation enabled using code data for validation of Airport names

A travelling order application has a screen with details of the departure airport, destination airport and the option to add multiple destinations.

The current system validates the airport abbreviations (such as LHR, NYC) but cannot identify airport name.

The requirement is that the user will be able to key in either the abbreviation or the airport name.

The design is to use an existing Code Data with all airports and IATA Airport Codes, and to add validation rules both for the airport abbreviation and airport name.

Three elementary processes were identified (order a flight, amend order, cancel order) using this code data validation. One nesting level and one DET are used for SP counting.

2: Data Entry validation enabled using logical checks on the DETS - adding an employee to an organization

In the processing logic of adding an employee to an organization, the number of validations performed while data entry on a screen and the complexity level of these validations, are considered non-functional.

Employee ID, in this example, is not generated automatically but manually entered besides employee name, department, date of birth and more. Validating that the set of data is correct is sized using this subcategory.

(These are considered to be technical requirements).

As per the CPM, some operations, which are performed in the elementary process to ensure valid data, are added into the data information. The data elements in these operations are not visible to the user (although agreed with the user) and not counted as DETs in FP. Data elements that are involved in these hidden operations should be counted using SNAP model.

See also Part 2 Chapter 1 example 1

1.2 Logical and Mathematical Operations

Definition

Extensive logical decisions, Boolean operations, and extensive mathematical operations applied on the process.

Extensive Mathematical Operations

SNAP defines an "extensive mathematical operation" as a mathematical operation which includes using one or more algorithms. An algorithm is defined for SNAP as a series of mathematical equations and calculations executed in conjunction with, or according to, logical operators to produce results identifiable to the user. Examples of extensive mathematical operations include using the Program Evaluation Review Technique (PERT) to calculate the expected completion date of a project, calculating the optimal profit for a business process using linear programming, determining the way to formulate the fastest flowing waiting lines using queuing theory, and finding the shortest route through a network. Examples of other algorithmic mathematical operations fitting the definition of "extensive" include solving calculus integration formulas, calculating federal income taxes, GPS calculations, gaming, weather forecasting, and perhaps calculating retirement pensions.

The DETs counted are the set of those required to operate the extensive mathematical operation, such as values for an algorithm's variables and settings maintained by the algorithm's control information. These values and settings are not necessarily stored in a single physical file; they may be stored in various locations such as settings of the value of variables located in the code or as DETs in various physical files. However located, as a set(s) this satisfies the requirements for either an internal logical file(s) or external interface file(s) because they are the logical grouping(s) of data necessary to operate the algorithm.

"Simple" or "routine" mathematical operations are defined here as not using algorithms. Examples can include adding a column of numbers, balancing a checking account, totaling daily sales, and calculating averages. Also, an application may require a simple or routine mathematical operation to be iterated many times. For example, a fast food restaurant manager may need to place an order for ketchup packets from a supplier. The manager first counts the current inventory of ketchup packets, forecasts the expected usage, and places an order to make up for the expected resulting shortfall. This is a simple or routine mathematical operation. If the manager has 100 types of items in inventory, and must perform this calculation 100 times to complete the total order with the supplier, then this is still defined as being a simple or routine mathematical operation because the simple or routine mathematical operation is iterated 100 times: "extensive" refers to the depth of the algorithm(s), not to the number of simple or routine calculation iterations needed."

Sizing an elementary process is determined by the type of processing logic

used by the external input (EI), EO, or external inquiry (EQ). While this can give a higher size to the elementary process that contains mathematical operations, it does not necessarily correlate to the effort needed to produce extensive mathematical operations. SNAP size compensates for the additional complexity of extensive mathematical operations.

Extensive Logical Operations

SNAP defines an "extensive logical operation" as a logical operation either containing a minimum of 4 nesting levels, containing more than 38 DETs required to operate the logical operation, or both. These DETs do not necessarily have to cross the application boundary.

The outcome of a logical operation may be a decision or set of decisions or evaluating a condition using data that exist in one or more logical files. The SCU is the elementary process. If more than one logical operation can be executed within the elementary process, then count either the combined number of DETS in the operations containing a minimum or 4 nesting levels, or count the sum of the DETs involved in all of the logical operations assuming that the sum is more than 38 (whichever is larger).

SCU

The elementary process.

Complexity Parameters:

1) FTR density of the logical file being accessed to do the business logic processing - table 1-8.
2) Processing logic type of the elementary process (logical / mathematical) – table 1-9.
3) Number of data elements types (#DETs) – table 1-10.
   a) FTR density Factor is measured as follows

	FTR density		
	0-3 FTR	**4-9 FTR**	**10+ FTR**
	Low	Average	High

Table 1-8 FTR Density, Logical and Mathematical Operations

   b) Type of the elementary process (logical / mathematical)

   Identify the type of the elementary process

EP Type	Main Purpose of the EP
Logical	Decision making or evaluating a condition using data that exist in one or more logical files (internal and / or external) Example: Exception processing
Mathematical	Transformation of data and / or use of control information that exist in one or more logical files (internal and / or external) that is used for an extensive mathematical operation. Example: Complex tax calculation

Table 1-9 EP type for Logical and Mathematical Operations

Note:

- When the main purpose cannot be clearly identified, select "Logical" (Do not count it as one Logical and one Mathematical).

SP calculation

Calculate SP based on the constant factor and the FTR density factor.

	Complexity Level		
	Low	**Average**	**High**
	SP=	**SP=**	**SP=**
EP type: Logical	4*#DETs	6*#DETs	10*#DETs
EP type: Mathematical	3*#DETs	4*#DETs	7*#DETs

Table 1-10 SNAP sizing for Logical and Mathematical Operations

Examples

- Project scheduling critical path analysis.
- Complex tax calculations.
- Linear programming algorithms.
- Calculus integration formulas.
- Financial return on investment calculations for a large industrial machine.
- Statistical Analysis of Variance calculations.
- Business sales forecasting using the ensemble forecasting method.

1.3 Data Formatting

Definition A requirement that deals with structure, format, or administrative information in a transaction not directly relevant to functionality that is seen by the user.

SCU The elementary process.

Complexity Parameters:

1. Transformation complexity:
    a. Low: Data type conversions or simple formatting such as byte padding, or data substitution, using a maximum of 2 operators (Celsius to Fahrenheit, Single Integer to Double Integer)
    b. Average: Involves encryption / decryption which is a characteristic of the application and applies to almost all processes, which is provided through a library -API interface
    c. High: Involves local Encryption/Decryption.
2. Number of data elements types (#DETs) transformed

Notes:

- Data elements refer to all types of data. See Part 1 Chapter 5 (Definitions of additional terms used in this chapter) for the definition of data elements
- Encryption algorithm is complex for- 1. Design specifically allowed to several key lengths; 2. Provide a method to ensure data integrity for high volume data; 3. Formatting medical images; 4. Restructure huge volume database etc.

SP calculation Identify the complexity based on transformation. Calculate SP based on the constant factor and the number of DETs (#DETs).

	Transformation Complexity		
	Low	**Average**	**High**
SP =	2*#DETs	3*#DETs	5*#DETs

Table 1-11 SNAP sizing for Data Formatting

Notes: It may include adherence to standards, layouts, messages.

Examples (See note below)

1. Simple transformations:
   - Convert text to value or value to text
   - Data formatting required for reporting requirements
   - An application shows the date (MMDDYYYY), time (GMT), current atmospheric temperature (degree Fahrenheit) in a standard format. However due to regulations, the date is required to be displayed as 'YYYYMMDD, the time should always show the local time zone, and temperature should be displayed as "Degrees Kelvin."
     As a result the display formats needs to be converted to adhere to the standards prescribed.
2. Complex transformations:
   - Enabling multi-lingual support for an application by using Code Data
   - Encryption/ Decryption, Compression - Decompression
   - Compliance to standards for electronic transfer of data in healthcare environment. The data packets are sent and received in particular format of EDI transactions.
     For example: Change the structure of the transactions - add headers and footers; the transaction format is changed per HIPAA (Health Insurance Portability and Accountability Act of 1996) requirements with no changes in the functionality.
   - Data interchange formats - XML to other formats, or other means of data interchange between two computer systems.
   - Preparation of metadata for various screen requirements or data warehouse views.
   - Transformations in data warehouse.

Note: When a transformation is agreed as functional between the user and the development team, and transformation is FP counted, do not add SP. If it is agreed as NFR between user and development team, use SP

See also Part 2, Chapter 1, example 2.

1.4 Internal Data Movements

Definition Data movement process from one partition to another within application Boundary with specific data handling. Data handling may include data formatting, logical /mathematical operations or Reference Data Maintenance.

SCU The elementary process, which crosses partitions. If an elementary process crosses more than one partition, use the formula below per each partition crossing (in figure 1-3, an elementary process may move from component 1 to component 2, and then to component 3. In such case, SP will be calculated at each partition crossing)

Complexity Parameters:

1. Number of data elements types (#DETs) transferred into and out of the partition, in which data is processed and / or maintained
2. Number of FTRs either read or updated by the elementary process

SP calculation Identify the complexity level based on the number of FTRs read/updated. Calculate SP as per the table below:

Identify the number of internal boundaries crossed and #DETs transferred. Calculate SP using the equation

	Complexity Level		
	Low (0-3 FTR)	**Average (4-9 FTR)**	**High (10+ FTR)**
SP =	4*#DETs	6*#DETs	10*#DETs

Table 1-12 SNAP sizing for Internal Data Movements

Notes: Internal Data Movements sub-category sizes transactions that do not cross the boundary of an application. These transactions are used in case of partition.

It may include:

- Data backup within application boundary, crossing partitions
- Data copy/movement between tables within application boundary, crossing partitions
- Realign data in temporary storage
- Transactions between application and middleware that do not cross functional boundaries
- SOA (Service Oriented Architecture) solutions. (When SOA functionality are within the application boundary)
- Data movements between tiers that do not cross functional

boundaries

- Data formatting transactions which use Code Data that crosses partition

Sub-Category 3.3 (Batch Processes) covers batch jobs. Batch jobs may be executed within a single partition.

- In case that there are application features like screens or consoles and such functionality is counted in IFPUG FPA, do not count it again using SNAP

Example See also Part 2, Chapter 1, example 3.

1.5 Delivering Added Value to Users by Data Configuration

Definition Additional unique business value to users that is provided by adding, changing or deleting reference data/ code data information from the database or data storage with no change in software code or the database structure.

SCU The elementary process per logical file

Notes: The SCU is the elementary process to consume the added value in the logical file and not the process to create or modify the configuration

Example: A new service is defined by adding its attributes to reference tables. The application is flexible enough to provide the new service with no code changes. Elementary processes to be counted may be: add the new service; change this service; cease the service. The process to add the service attributes to the reference tables (i.e. writing and using scripts) should not be counted.

In case the configured data impacts several elementary processes, each elementary process is counted separately

Terms

Attribute

An independent parameter that has a unique business meaning and contains a set of different values

A record

One row in a logical file

A Logical File

A user recognizable group of logically related data or control information.

Complexity Parameters:

1. Number of unique attributes involved in the elementary process, that are added / modified / deleted

2. Number of Records configured

SP calculation

Identify the complexity level based on #Records. Calculate SP based on the constant factor and the #Attributes.

	Complexity Level		
	Low	**Average**	**High**
	1-10 records	**11-29 records**	**30+ records**
SP =	6*#attributes	8*#attributes	12*#attributes

Table 1-13 SNAP sizing for Data Configuration

Notes:

New services, products, price plans etc., can be added to the application by adding or changing reference data and not by writing code.

Functionality by Data Configuration brings added value to the user, also adds effort to configure and test the new functionality.

Examples

This sub-category sizes functionality that is created by adding data to the database.

1. Application requires granting access to specific role in the application. To meet this requirement, developer does not write any separate code and instead updates a configuration file, and associates the user or set of users into some property file(s). Such additions or changes are made to meet user requirements, which affect the functionality at the elementary process level.

The process to configure the data into the database is not sized separately. Only the user's processes should be counted.

2. Application requires configuring a new product ("Product X here below) or a component that can be sold using the application. The new product and its price plan are defined in reference data. The project effort may be creating the data, by migrating it to the reference files and testing that the application functions with the new data. The assessment identifies many SCUs here.

- Change product Y to product X.
- Provide product X.
- Change price of product X etc.

See also Part 2, Chapter 1, example 4.

Category 2: Interface Design

Interface Design

The Interface Design Category relates to the end user experience. This category assesses the design of UI processes and methods that allow the user to interface with the application.

2.1 User Interfaces

Definition

Unique, user identifiable, independent graphical user interface elements added or configured on the user interface that do not change the functionality of the system but affect non-functional characteristics (such as usability, ease of learning, attractiveness, accessibility)

SCU

Set of screens as defined by the elementary process

Terms

UI Element

UI Element (User Interface Element) is a unique user identifiable structural element which makes up the user interface. It includes elements such as:

1) Window (which can be container, child, text or message window)
2) Menus
3) Icons
4) Controls
    a. Pointer (or mouse cursor)
    b. Text box
    c. Button
    d. Hyperlink
    e. Drop-down list
    f. List box
    g. Combo box
    h. Check box
    i. Radio button
    j. Cycle button
    k. Datagrid
5) Tabs
6) Interaction elements like a cursor
7) Labels

The above controls are used to display or manipulate data objects. The aspect that adds to complexity of the design, configuration and testing time of a user interface is the configuration of each of these elements. Non-functional requirements may involve changing the properties of these UI Elements. Depending upon the type of the user-interface element, varying number of properties can be configured to produce a desired output. For Example, Button could be set to “locked” or

"highlighted" or "colored" or placed at a particular location on the screen.

UI Element Properties

Each UI Element is associated with certain properties which define the behavior and look and feel of the User Interface Element. For example, a window would have properties like: background color, active border, active caption etc.

A button can have properties like: ButtonHighlight, ButtonText, BackgroundColor etc.

Tool Tips can have properties like: Info Text, Info Background etc.

The list of properties for several more User Interface elements can go on and the above examples have been cited from:

www.w3.org/TR/CSS2/ui.html

UI Element Set

A UI element set is the collection of all the UI elements of the same type in the SCU.

Example: All the text boxes in the set of screens (SCU).

Complexity Parameters:

1. The sum of the number of unique properties configured for each UI element in the SCU.
2. Number of unique UI elements impacted.

SP calculation

Identify the complexity based on number of properties of UI element set. Calculate SP as the product of the constant factor and the number of unique UI elements.

	UI Type Complexity		
	Low	**Average**	**High**
	<10 properties added or configured	**10 - 15 Properties added or configured**	**16+ Properties added or configured.**
SP =	2*# unique UI elements	3 * # unique UI elements	4*#unique UI elements

Table 1-14 SNAP sizing for User Interfaces

Rules

1. If the process for adding/changing of UI is FP counted, then do not duplicate the count in the assessment; however, changing the contents and appearance of a GUI element needs to be assessed as non-functional. Aesthetic changes in UI screen, Static or Dynamic UI Pages, Rearranging of screen and printed reports should be assessed

under User Interfaces sub-category.

2. The sets of screens within one process will be counted as the SCU

Notes:

UI Elements added may be of any form, such as text, sound, picture, logos, color, keys, controls, navigation, animating or enabling/disabling the above (when such enabling/disabling is not functional)

Added operations that support: Function Keys; Auto fill; Short cut keys; Common Keys; Navigation - Screen/Page level

Example

Some text on users' screens is hard coded. Due to a new policy of the company, the request is to replace the word "customer" with the words "Business partner."

The analysis found that the word "customer" should be replaced in the following UI elements (Note: since SNAP counts the number of unique UI elements, there is no need to estimate the number of occurrences of each unique UI element)

SCU 1: Acquire a new business partner:

Header, labels, radio button, drop-down list

SCU 2: Modify details of a business partner:

Header, labels

SCU 3: Send message to a business partner:

Header, labels,

SCU 4: Cease services to a business partner:

Header, labels

Changing the text in these UI elements is considered one property. UI type complexity is Low

SP = 2*# unique UI elements per each SCU:

SP = 2*(4 + 2 + 2 + 2) = 20 SP

See also Part 2, Chapter 1, example 5.

2.2 Help Methods

Definition Information provided to the users that explains how the software provides its functionality or other supportive information provided to users.

SCU The assessed application.

Terms

Help item

A Help Item is the smallest, unique, user identifiable help or information topic which provides the user supportive information or details about a particular part of software.

Context Help

Context Help refers to a help program feature that changes depending on what user is doing in the program. It is a kind of online help that is obtained from a specific point in the state of the software, providing help for the situation that is associated with that state.

Context-sensitive help can be implemented using tooltips, which either provide a brief description of a GUI widget or display a complete topic from the help file. Other commonly used ways to access context-sensitive help start by clicking a button. One way uses a per widget button that displays the help immediately. Another way changes the pointer shape to a question mark, and then, after the user clicks a widget, the help appears.

Static Web Page

A static web page is a web page that is delivered to all the users exactly as stored, displaying the same information to all users, and is not generated by an application.

Complexity Parameters:

1. Help Type
   a. User Manual (Hard copy/ soft copy / Application level help)
   b. On-line Text
   c. Context
   d. Context + On-line
2. Number of help items impacted

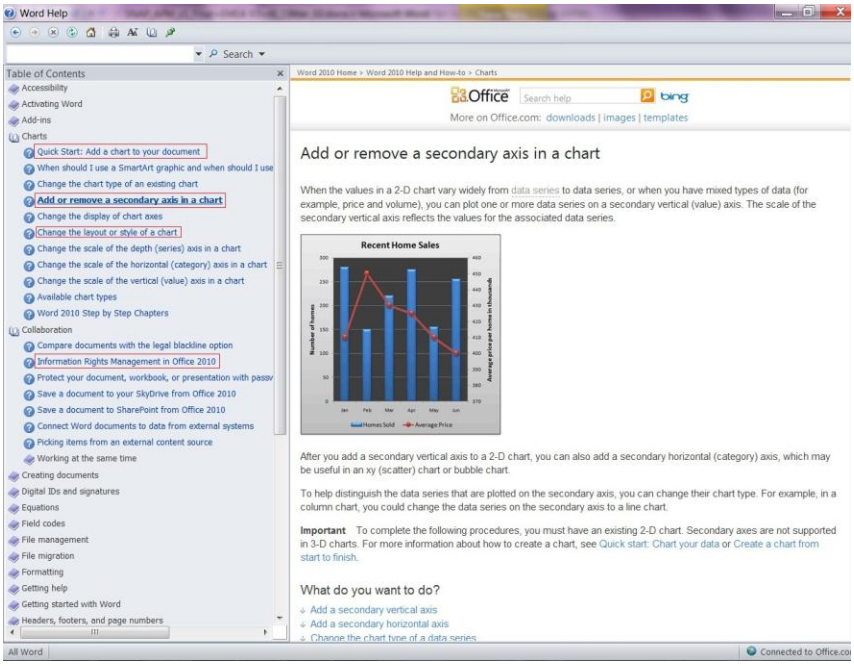
SP calculation Identify the help type. Calculate SP based on the constant factor and the number of help items impacted.

Help Type	SP =
User Manual	1*(#help items)
On-line Text	2*(#help items)
Context Help	2*(#help items)
Context + On-line	3*(#help items)

Table 1-15 SNAP sizing for Help Methods

Example

A good example of Help Item can be given from Windows Help. When we click F1 in MS Word/Excel, a help window appears and we can see a Table of Contents in the left. By clicking each of the Help Content items, we can see the granular level of help which has the details of the help sub-topic. Each of these smallest granular help topics can be an individual Help Item. A screenshot showing the same is given below. The Help Items are encircled in Red.



See also Part 2, Chapter 1, example 6.

Notes:

Static web pages

Use the above explanation to size static web pages. Although a static page is not directly a "help item", the Help methods subcategory should be used to identify the complexity' and then calculate the non-functional size.

Although there may be UI aspects in building and maintaining static web pages, do not add size using sub-category 2.1. The primary intent of 2.1 is to address GUI changes to improve usability, look and feel, learnability etc. of the functionality of the application

Help items that involve User Interfaces

There are many cases in which adding a Help item involves UI effort. In such cases, it is not expected to size this activity twice, as SP for Help sub-category and additional SP for User Interfaces.

When the primary intent of the activity is creating a Help item, only this sub-category should be used

The calibration of the equations in this sub-category remains open for future research based on pilot data from industry.

2.3 Multiple Input Methods

Definition The ability of the application to provide its functionality while accepting multiple input methods.

SCU The elementary process.

Terms

Input Method

A technique or media type, which is used to deliver data into the assessed application, such as bar code reader, fax, PDF, office document, screen, voice message, SMS, smart mobile device etc.

The assessed application may need to identify the input method in order to interpret and use the received information.

Complexity Parameters:

1. The number of data element types (DET's) in the SCU.
2. The number of additional input methods.

SP calculation Identify the complexity based on the number of DET's. Calculate SP based on the constant factor and the number input methods.

	Input Methods Complexity		
	Low	**Average**	**High**
	1-4 DET's	**5-15 DET's**	**16+ DET's**
SP=	3*# additional input methods	4*# additional input methods	6*# additional input methods

Table 1-16 SNAP sizing for Multiple Input Methods

Rules This category should be used to when there are multiple types of inputs used to invoke the same functionality. If the different input types differ in terms of DETs, FTRs and processing logic, then they would already have been accounted as separate functions in function point counting process.

If they are same, then multiple input methods should be used.

Check the following

1) Approach taken for FP counting - single instance or multiple instance
2) The multiple methods of input for the same functionality (Same DETs, FTRS and processing logic) have not been included for FP size calculation. In other words, if the FP count has been done using single instance approach for different media types, then the additional input method of same data entry needs to be accounted for using SNAP. For example the same input can be provided via a smart phone or a web screen.
3) If multiple input methods are already accounted for in the FP count or the multiple instance approach has been taken for FP counting, then it

should be excluded from the SNAP assessment.

Example See also Part 2 Chapter 1 example 7

2.4 Multiple Output Methods

Definition The ability of the application to provide its functionality while using multiple output methods.

SCU The elementary process

Terms

Output Method

A technique or media type, which is used to deliver data from the assessed application, such as fax, PDF, office document, screen, voice message, SMS etc.

The assessed application may need to manipulate the sent information in order to send it to the various outputs.

Complexity Parameters:

1. The number of DATA Element Types (DET's) in the SCU
2. The number of additional output methods

SP calculation Identify the complexity based on the number of DET's. Calculate SP based on the constant factor and the number output methods.

	Output Methods complexity		
	Low	**Average**	**High**
	1-5 DET's	**6-19 DET's**	**20+ DET's**
SP =	3*# additional output methods	4*# additional output methods	6*# additional output methods

Table 1-17 SNAP sizing for Multiple Output Methods

Rules This category should be used to when there are multiple types of outputs used for the same functionality. If the different output types vary in terms of DETs, FTRs and processing logic, then they would already have been counted as separate functions in function point counting process.

If they are same, then multiple output methods should be used.

Check the following

1) Approach taken for FP counting – single instance or multiple instance
2) The multiple methods of output for the same functionality (same DETs, FTRS and processing logic) have not been included for FP size calculation. In other words, if the FP count has been done using single instance approach for different media types, then the additional output method of same data entry needs to be accounted for using SNAP.

   For example the same output can be provided to a smart phone or to a

web screen.

3) If multiple output methods are already accounted for in the FP count or the multiple instance approach has been taken for FP counting, then it should be excluded from the SNAP assessment.

Example See also Part 2, Chapter 1, example 7.

Category 3: Technical Environment

Technical Environment

The Technical Environment category relates to aspects of the environment where the application resides. It assesses technology as well as changes to internal data and configuration that do not provide added or changed functionality from a Function Points perspective.

3.1 Multiple Platforms

Definition

Operations that are provided to support the ability of the software to work on more than one platform.

Note:

In order for software to be considered multi-platform, it must be able to function on more than one computer architecture or operating system. This can be a time-consuming task given that different operating systems have different application programming interfaces or APIs (for example, Linux uses a different API for application software than Windows does).

SCU

The elementary process

Terms:

Computing platform includes a hardware architecture and a software framework (including application frameworks), where the combination allows software, particularly applications software, to run. Typical platforms include a computer's architecture, operating system, programming languages and related user interface (run-time system libraries or graphical user interface).

Software Platforms: Software Platform is a framework used for the software application development. Different programming languages can be grouped into several platforms based on the programming language family.

A programming language can belong to a particular software language family like Object Oriented, Procedural, Declarative etc.

Object Oriented: Java, C++, C#, Javascript, Phython, Smaltalk, VB, VB.NET etc

Procedural: C, Fortran, PHP, COBOL etc

Declarative: SQL, XQuery, BPEL, XSLT, XML etc

Hardware Platforms: A hardware platform can refer to a computer's architecture or processor architecture. For example, the x86 and x86-64 CPUs make up one of the most common computer architectures in use in general-purpose home computers.

Complexity Parameters:

1. Nature of platforms (i.e. software, hardware)
2. Numbers of platforms to operate

SP calculation

Identify the different software and hardware platforms involved and the number of platforms to operate. Calculate SP based on the platform category row from the table below and the number of platforms. If more than one row is applicable, then SP is sum of constant factors obtained from each category applicable.

	SP =		
	2 platforms	3 platforms	4+ platforms
Category 1 : Software Platforms: Same Software Family	20	30	40
Category 2: Software Platforms: Different Family	40	60	80
Category 3: Software Platforms: Different Browsers	10	20	30
Category 4: Hardware Platforms: Real time embedded systems	TBD*	TBD*	TBD*
Category 5: Hardware Platforms: Non-Real time embedded systems	TBD*	TBD*	TBD*
Category 6: Combination of Hardware and Software: Non-Real time embedded systems	TBD*	TBD*	TBD*

Table 1-18 SNAP sizing for Multiple Platforms

Calculation example

If an application is built on JAVA and COBOL and requires multiple (more than 4) browser support, then SP would be

40 SP per each SCU that is built on both Java and Cobol, (java and COBOL are considered as different family

plus

30 SP per each SCU that is to work on multiple browsers

*TBD: To be Defined

Example

- Software Platform: .NET, Java
- Operating System Platform: MS Windows, Linux, IBM/Microsoft Operating System 2, Mac OS
- Hardware Platform: Mainframe computers, Midrange computers, RISC processors, Mobile device architecture, Mac systems.
- Browsers: Internet Explorer, Firefox, Google Chrome, etc.

See also Part 2, Chapter 1, example 8.

Note: Currently the platforms considered in the calibration of the model are only software type platforms.

Please note that this category should be used only if same set of functionality is being delivered on multiple platforms. This is the case where business functionality is the same but it needs to be delivered in two different environments. For example, same application functions are built on JAVA and also on VC++ to suit client requirements, and then this category can be used.

If the architectural framework itself consists of different platforms to deliver part of functionality, then this category should not be used. This is a usual case where different technical components interact with each other to deliver application functions. No duplication of effort takes place to rebuild the same functionality in different environment.

3.2 Database Technology

Definition Features and operations that are added to the database or to the statements to read / write data to and from the database to deliver non-functional requirements without affecting the functionality that is provided

SCU The elementary process

Terms

Database Changes

Each of the following is considered as one change.

1. A Business table or a Reference table change or table creation, such as:
    a. Adding tables or adding columns for non-functional purposes only.
    b. Rearranging the order of column in a table.
    c. Changing or adding relationships using referential integrity features.
    d. Changing the primary key without dropping and adding the primary key.
2. Creation of Code Data table or Updates to code tables
    a. Adding tables or adding columns for non-functional purposes only.
    b. Rearranging the order of column in a table.
    c. Changing the primary key without dropping and adding the primary key.
3. Updates of data in Code Tables.
4. An indexing change, such as:
    a. Changing the columns used for indexing.
    b. Changing the uniqueness specification of an index.
    c. Clustering the table data by a different index.
    d. Changing the order of an index (ascending or descending).
5. A change of Database views (see definition in Part 1 chapter 5) and partitions, such as:
    a. Changing or adding database partitioning.
    b. Adding, changing or removing a Database view.
6. A Database capacity change, such as:
    a. Tables space.
    b. Enhancing the performance features .

7. A Query / insert change, such as:
    a. Changes to queries or data selection or inserts to the database without adding, deleting or changing functionality.

   For example, Changing the primary key and adding relationship is counted as one change.

Complexity Parameters:

1. Logical File complexity.
2. The number of database-related changes.

Changes to the database might be done for any non-functional requirement such as performance, capacity management, data integrity etc. Complexity of implementing any such change would depend on the complexity of the Logical File as well as the # of changes.

a. Logical File Complexity Factor.

		DETs		
		1-19	**20-50**	**>50**
RETs	**1**	Low	Low	Average
	2-5	Low	Average	High
	>5	Average	High	High

Table 1-19 Logical File Complexity, Database Technology

SP calculation

	FTR Complexity Factor		
	Low	**Average**	**High**
SP=	6* # of changes	9* # of changes	12* # of changes

Table 1-20 SP calculation, Database Technology

If there are multiple FTRs being impacted for the NFR which are all impacting the same EP, then the higher complexity of the FTR should only be considered as the Complexity Factor not the individual FTRs separately.

Example

An EP "Create Order" is designed for performance improvement. To achieve this, a "read only" database view is created on "Customer" FTR having 18 DET and 3 RET (FTR complexity is "Low")

In addition, an index is created on "Order Placed" FTR having 30 DET and 3 RET (FTR complexity is "Average").

The highest FTR complexity is "Average"; therefore, for the two changes,

SP=9*2 = 18

See also Part 2 Chapter 1 examples 3 and 9

3.3 Batch Processes

Definition

Batch jobs that are not considered as functional requirements (they do not qualify as a transactional function) can be considered in SNAP. This sub-category allows for the sizing of batch processes which are triggered within the boundary of the application, not resulting in any data crossing the boundary.

Non-functional requirements (NFR) associated with batch jobs such as improving the job completion time, increasing the capacity of the job to process higher volumes of transactions, or performance improvement requirements may be sized using other SNAP sub-categories as applicable (3.2, 1.1 or 1.2).

However, if a NFR related to batch processing is not covered under these sub-categories, it may be considered in 3.3.

SCU

User identified batch job

Note:

When several batch jobs are automated (run always as a whole) and only the end result is user identifiable, count these batch jobs as an individual SCU

Complexity Parameters:

1. Number of DETs processed by the job.
2. Number of FTRs either read or updated by the job.

SP calculation

For each job calculate SP as:

Identify the complexity level based on the number of FTRs read/updated. Calculate SP as per the table below:

	Complexity Level		
	Low (1-3 FTR)	**Average (4-9 FTR)**	**High (10+ FTR)**
SP =	4*#DETs	6*#DETs	10*#DETs

Table 1-21 SNAP sizing for Batch Processes

Notes:

User specified one-time data loads to logical tables, can be counted using this category. Please note that these data loads should not be migration related data loads, which are counted as Conversion FP using function points.

Example

- Different processes are merged to one batch: Count the DETs and FTRs in the merged batch.

- Intermediate data for job validation that are in Code Data.
- Scheduler data instructs to perform subsequent process steps, which are in Code Data.

See also Part 2, Chapter 1, example 10.

Category 4: Architecture

Architecture The Architecture Category relates to the design and coding techniques utilized to build and enhance the application. It assesses the complexities of modular and/or component based development.

4.1 Component Based Software

Definition Pieces of software used within the boundary of the assessed application to integrate with previously-existing software or to build components in the system.

SCU The elementary process.

Terms

A Software Component

A piece of software offering a predefined service and which is able to communicate with other components via standard interfaces.
An individual software component is a software package, a Web service, or a module that encapsulates a set of related functions (or data). The essence of a "component" is the encapsulation of business logic or technical functionality which admits a standard interface. Software component is the element that conforms to a component model and can be independently deployed and composed without modification, according to a composition standard. A component model defines specific interaction and composition standards. A component model implementation is the dedicated set of executable software elements required to support the execution of components that conform to the model.

Criteria for software components:

1. Performs a specific functionality.
2. Capable of Parallel execution: multiple-use.
3. Exchangeable: Non-context-specific.
4. Composable with other components (can be selected and assembled in various combinations to satisfy specific user requirements).
5. Encapsulated i.e., non-investigable through its interfaces.
6. A unit of independent deployment and versioning with well-defined interfaces and communicates via interfaces only.
7. Has a structure and behavior that conforms to a component model like .COM, CORBA, SUN Java etc.

Examples Below picture shows simple components interacting with each other (source: Wikipedia).

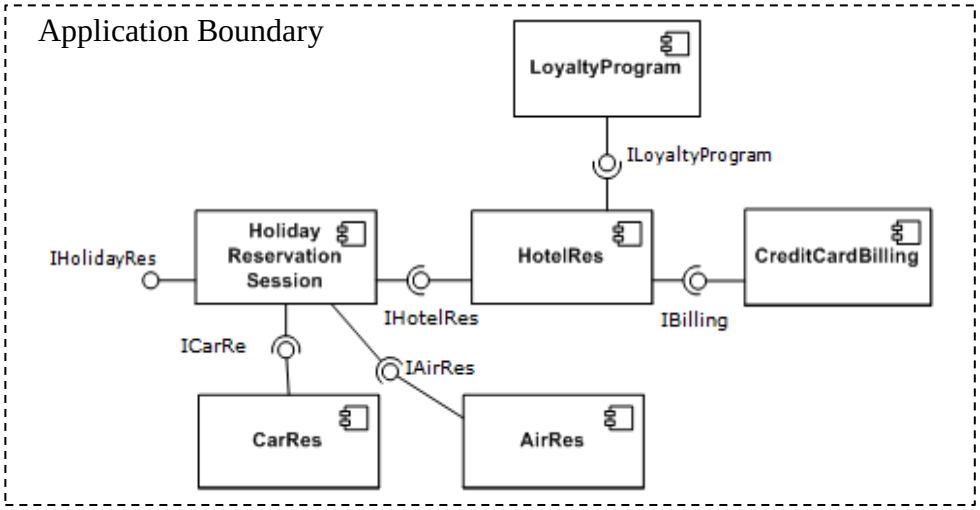


Figure 1-5 Components model for holiday reservation system

Complexity Parameters:

1. Third-party component or in-house reuse.
2. Number of unique components that are involved in the elementary process.

SP calculation Calculate SP based on the constant factor and the number of unique components.

Type	SP Calculation
In-house components	SP=3*(# of unique components)
Third party components	SP=4*(# of unique components)

Table 1-22 SNAP sizing for Component based software

Example See Part 2, Chapter 1, example 11.

Notes: This sub-category does not size the functionality of the component. Follow the instructions of the CPM to count the functional size of the components.

Reuse of components may be applied to meet non-functional requirements such as Maintainability ("The capability of the software product to adhere to standards or conventions relating to maintainability"), changeability, maturity or replaceability.

4.2 Multiple Input / Output Interfaces

Definition

Applications required supporting multiple input and output interfaces (user files with the same format) are covered in this subcategory. For example: due to a growing number of users and volume of data over a period of time.

Adding more input/output interfaces without changing the functionality is not considered functional change and hence such changes are not sized by FP. This sub-category should be used to size such changes in an application.

Note:

If the project/organization considers adding new input/output interfaces as a functional change, then function points would be used for sizing, and SNAP should not be used.

SCU

The elementary process.

Complexity Parameters:

1. The number of data element types (DET's) in the SCU.
2. The number of additional input and output interfaces.

Notes:

Count the number of additional input and output interfaces.

When an interface is used for both input and output, count it once as an input and once as an output.

SP calculation

Identify the complexity based on number of DETs.

The SP is the product of the factor derived from the number of DETs specified in the table below and the number of added interfaces.

	Complexity Level		
	Low	**Average**	**High**
	1-5 DETs	**6-19 DETs**	**20+ DETs**
SP =	3* (Additional # of Interfaces)	4* (Additional # of Interfaces)	6* (Additional # of Interfaces)

Table 1-23 SNAP sizing for Multiple Input /Output Interfaces

Example An example illustration is provided below:

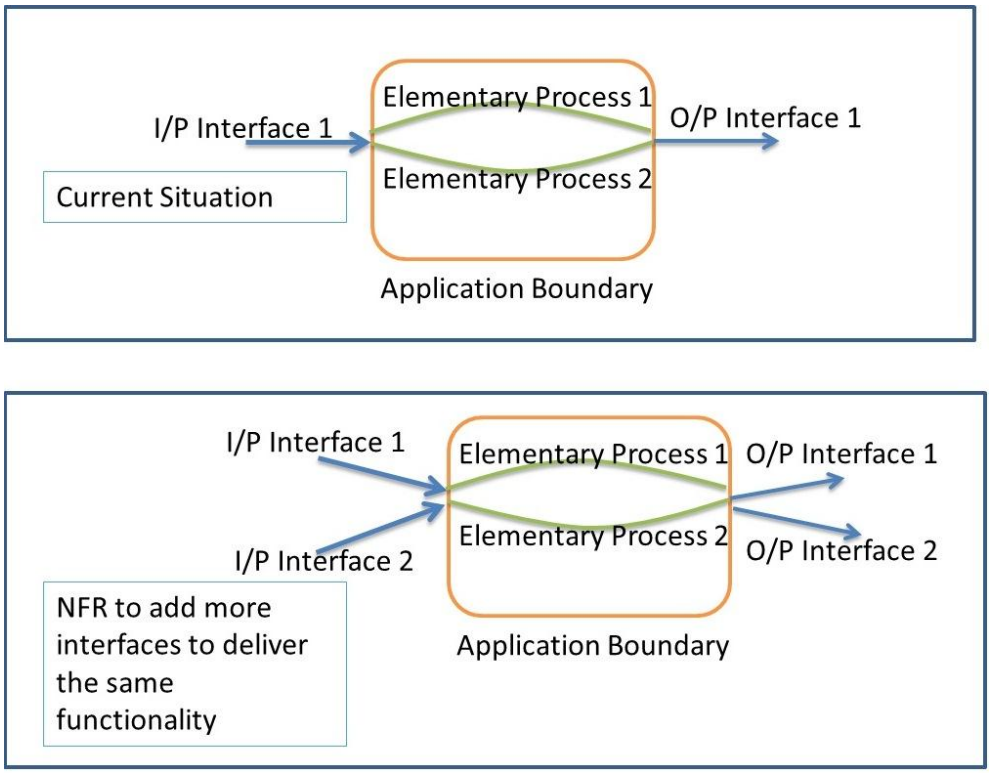


Figure 1-6 – Multiple interfaces

Note A key difference between this sub-category and 2.3 / 2.4 multiple input /output methods is that in 4.2, the existing interface is replicated with the same technology to give all users the same level of performance and same experience.

This sub-category is not related to graphical user interfaces

See also Part 2, Chapter 1, example 12.

Mission Critical/Real Time Systems

Introduction

Some software systems might be tagged as Real Time or Mission Critical based on the timeliness or accuracy or severity of the consequences associated with their output.

Terms

Real Time

Real Time software is software which fails if a timing deadline is not met.

Mission Critical

Mission Critical software is a software whose failure might cause catastrophic consequences (such as someone dying, damage to property, severe financial losses, etc.)

The timeliness, accuracy, high throughput aspects of such transactions might be considered part of the functional aspect as these characteristics are the sole basis of classifying a system as Real Time/Mission Critical.

However in case the timeliness, accuracy, high throughput aspects are considered as non-functional requirements, their sizing may be done using some of the other sub-categories.

Examples:

Timeliness might be achieved by tuning the database interaction transactions or making database changes or combination of both for improved performance. In such a case, subcategory 3.2 will be used for SNAP calculation.

SP calculation

Accuracy might be achieved by adding more validations and logical/mathematical operations. Subcategory 1.1, 1.2 might be used for SNAP calculation.

Higher Throughput might be achieved by splitting existing transactions to process multiple inputs in parallel. Subcategory 2.3 might be used for SNAP calculation.

The solution mentioned for the above three scenarios is not exhaustive. If some other approach is used to meet such requirements the appropriate sub-category may be used.

Hence Real Time or Mission Critical transaction/system should be assessed by using the other sub categories.

SP Calculation Example

The following table demonstrates how SNAP size is calculated for the "Data Operations" category.

The complexity is calculated according to the formulas presented in this Chapter.

		Category 1: Data Operations														
	Sub-Category:	1.1 Data Entry Validation				Data Entry Validation SNAP Count	1.2 Logical and Mathematical Operations					Logical and Mathematical Operations SNAP Count	1.3 Data Formatting			Data Formatting SNAP Count
	SCU Definition:	SCU=The Functional Elementary Process					SCU=The Functional Elementary Process						SCU=The Functional Elementary Process			
No.	SCU Description / ID	Number of nesting levels	Complexity Level	Number of DETs	Formula		Elementary Process Type (Main Purpose)	Number of DETs	Number of FTR	Complexity Level	Formula		Transformation Complexity Level (L, A, H)	Number of DETs	Formula	
1	Costomer orders a product	2	Low	18	=2*# DETs	36	Logical	4	1	Low	=4*# DETs	16	Low	4	=2*# DETs	8
2	Customer amends an order															
3	Invoice details changes	4	Average	15	=3*# DETs	45	Mathematical	12	2	Low	=3*# DETs	36				
4	Validate address	6	High	9	=4*# DETs	36										

	Sub-Category:		1.4 Internal Data Movements				Internal Data Movements SNAP Count	1.5 Delivering Added Value to Users by Data Configuration			Delivering Functionality by Data Configuration SNAP Count	Total SP for Data Operations:
	SCU Definition:	SCU	SCU=The Functional Elementary Process					SCU=The Functional Elementary Process				
No.	SCU Description / ID	Number of nesting levels	Number of FTRs	Complexity Level	Number of DETs	Formula		Number of Records	Complexity Level	Number of Attributes		
1	Costomer orders a product	2	3	Low	18	=4*# DETs	72	9	Low	7	21	153
2	Customer amends an order							9	Low	7	21	21
3	Invoice details changes	4										60
4	Validate address	6	1	Low	4	=4*# DETs	16					43

Table 1-24 SP Calculation Example

Calculate Non-functional Size (SNAP Points)

Introduction This section defines how to calculate the non-functional size of the project/product in scope.

Contents This chapter includes the following:

Formula Approach

During an assessment of a non-functional requirement, one or many of the categories can be assessed depending on the specification of the requirement. In addition, one or many of the sub-categories within the categories can be assessed depending on the specification of the requirement

For each non-functional requirement, it is possible to determine the non-functional size in four steps:

Step 1 For each requirement, identify the categories and sub-categories that are associated with the requirement.

Step 2 For each of the sub-categories, identify the SCUs

Step 3 Determine the non-functional size (SNAP Points -SP) for each SCU within the sub-category, by using the equation or the table for the sub-categories.

Step 4 Determine the SP for a specific project or application by using the formula for the project type in question.

Determine the Non-functional Size of Each Sub-category

The non-functional size of each sub-category shall be determined using the defined measure for the SCU for each sub-category.

There is one definition of the SCU for each of the sub-categories. These assessment criteria are defined in the sub-category definition in Chapter 2.

The SP for each sub-category is determined by using the defined equation or table for each sub-category.

Determine the Non-functional Size of Each Category

The SP is determined by using the equation for each category against the identified SP for each sub-category under the current category.

Formula for Category Data Operations

SP for Data Operations = [∑ of SP Data Entry Validations]
+ [∑ of SP for Logical and Mathematical Operations]
+ [∑ of SP for Data Formatting]
+ [∑ of SP for Internal Data Movements]
+ [∑ of SP for Delivering Functionality by Data Configuration]

Formula for Category Interface Design

SP for Interface Design = [∑ of SP for User Interfaces]
+ [∑ of SP for Help Methods]
+ [∑ of SP for Multiple Input Methods]
+ [∑ of SP for Multiple Output Formats]

Formula for Category Technical Environment

SP for Technical Environment = [∑ of SP for Multiple Platform]
+ [∑ of SP for Database Technology]
+ [∑ of SP for Batch Processes]

Formula for Category Architecture

SP for Architecture = [∑ of SP for Component Based Software]
+ [∑ of SP for Multiple Input / Output Interfaces]

Note: ∑ means "Sum"

Determine the Non-functional Size for a Development Project

The size of the non-functional requirements is equal to the sum of SP sizes of each category.

A development project non-functional size shall be calculated using the development formula

Formula for development project

DSP = ADD

where

DSP is the development project SP;

ADD is the size of the non-functional requirements delivered to the user by the development project.

ADD =

[∑ of SP for Data Operations] + [∑ of SP for Interface Design]
+[∑ of SP for Technical Environment] + [∑ of SP for Architecture]

The application non-functional size is equivalent to the non-functional size of the development project.

Note: For non-functional size converted functionality is not identified.

Determine the Non-functional Size for an Enhancement Project

Enhancement projects can involve additions, changes to, and deletion of existing non-functional features.

Enhancement project is a project to develop and deliver maintenance. It may be adaptive, preventive or perfective type

The enhancement project non-functional size is a measure of the non-functional requirements added, changed or deleted at the completion of an enhancement project as measured by the enhancement project SP.

Rules

Enhancement non-functional requirements shall be measured in accordance with the following:

a) Do not modify the boundary or partition already established for the

application(s) being modified

b) Assess requirements that are added, changed or deleted

c) The application non-functional size may be updated to reflect:

1) Added non-functional requirements, which increase the application non-functional size.

2) Changed non-functional requirements, which may increase, decrease or have no effect on the application non-functional size.

3) Deleted non-functional requirements, which decrease the application non-functional size.

An enhancement project non-functional size shall be calculated using the formula:

Formula for enhancement project

ESP = ADD+ CHGA + DEL

where

ESP is the enhancement project SP.

ADD is the size of the non-functional requirements being added by the enhancement project.

ADD =

[∑ of SP for added Data Operations] + [∑ of SP for added Interface Design] + [∑ of SP for added Technical Environment] + [∑ of SP for added Architecture]

CHGA is the size of the non-functional requirements being changed by the enhancement project – as they are/will be after implementation;

CHGA =

[∑ of SP for changed Data Operations] + [∑ of SP for changed Interface Design] + [∑ of SP for changed Technical Environment] + [∑ of SP for changed Architecture]

DEL is the size of the non-functional requirements deleted by the enhancement project.

DEL =

[∑ of SP for deleted Data Operations] + [∑ of SP for deleted Interface Design] + [∑ of SP for deleted Technical Environment] + [∑ of SP for deleted Architecture]

An application non-functional size after an enhancement project shall be calculated using the formula:

Formula for application after enhancement project

ASPA = ASPB + (ADD + CHGA) - (CHGB + DEL)

Where:

ASPA is the application SP after the enhancement project.

ASPB is the application SP before the enhancement project.

ADD is the size of the non-functional requirements being added by the enhancement project.

CHGA is the size of the non-functional requirements being changed by the enhancement project – as they are/will be after implementation.

CHGB is the size of the non-functional requirements being changed by the enhancement project – as they are/were before the project commenced.

DEL is the size of the non-functional requirements deleted by the enhancement project.

SNAP Calculation Case Study

Project scope

'StarTrek' is the code name for an enhancement project that the 'Alpha' team was commissioned to deliver. Being a relatively small project (in terms of budget), StarTrek had requirements to enhance the performance of the company's flagship application, 'Centra-One', along with minor changes to some of its existing functionalities.

SNAP Meeting

The brainstorming sessions held between the project's Chief architect, the Lead developer and the client resulted in the following set of requirements:

In order to boost the performance of the "invoice update" transaction (current average response time: 8-12 seconds), and bring the response time down under 3 seconds, a multi-pronged strategy was adopted.

Requirements Relevant for SNAP

Requirements relevant for SNAP are identified and reviewed

Requirement 1:

Create an additional interface for receiving "update invoice" transactions – Automated Fax interface (to divert a portion of the web update transactions and lessen the load that the server was facing). A separate server was installed to handle fax requests and was equipped to read and decode the incoming fax forms which were in a predetermined format.

Requirement 2:

a) Modify the validation logic of the current 'update invoice' online transaction to reduce its processing time.
The new validation needs one additional DET. This is considered as a functional change, generating a High EI (3 FTRs, 10 DETs)
(The validation is considered as a non-functional change)

b) Create a cache file that stores the list of most-commonly accessed customer records, which was internally referenced before updating the invoice. This, it was predicted, will result in further improvement in performance. This cache was refreshed at regular intervals.

c) Make some cosmetic changes to the 'update invoice' online screen and the 'home screen' to inform the users of the new setup.

The Alpha team wanted to size these requirements, to provide as an input to the estimation process and also to track the size change over the life cycle to indicate scope-creep. They naturally decide to use SNAP together with IFPUG FP as that allowed them the capability to size all functional and non-functional aspects of the project.

Purpose & Scope

Purpose of the sizing: to size the enhancement project and use it for project estimation and to track scope creep.

Type of count: Enhancement

Scope: Requirements 1 & 2 (a, b & c)

Boundary

SNAP assessment will assume the same boundary as that used by the IFPUG FP methodology. Though the Centra-One application had 3 layers in itself, the boundary was fixed at the same level as that for the FP methodology.

Assessment

During the high level design phase, the Alpha team did an impact analysis and found that the following SCUs would be impacted. The SCUs were determined as per the rules of SNAP.

1. Identify sub-categories
2. Identify the SCUs
3. Per sub-category, assess the complexity of each SCU, and calculate SNAP score.

Notes:

One requirement that includes both functional and non-functional aspects may impact both function points and SNAP (See 2a above)

One requirement may impact more than one elementary process and one SCU can be impacted by more than one requirement (functional/non-functional). While assessing, care must be taken to ensure that the SCUs are not double assessed in SNAP.

The definition of SCUs according to SNAP may be identical, for some sub-categories, to the traditional IFPUG elementary process. In other cases, sub-categories are assessed within different counting units (SCUs) level.

In addition to IFPUG FP counting, which has determined a count of 6 FP to this project, SNAP assesses it as 200 SP (38+34+31+41+24+32) based on the analysis in table 1-25 below.

The Alpha team used both the functional size (6 FP) and the non-functional size (200 SP) as input into their estimation process.

Req. #	*SCU Type*	Impacted SCU	FP counted	# of FP	SNAP counted	1.1 Data Entry Validation	1.3 Data Formatting	2.1 User Interfaces	2.3 Multiple Input methods	3.2 Database Technology
1	Elementary process	Update invoice	No	0	Yes	-	30	-	4	-
2a	Elementary process	Update invoice	Yes	6	Yes	8	-	-	-	-
2b	Elementary process	1. View customer 2. Update customer details	No	0	Yes	-	-	-	-	18
2c	Elementary process	1. Updates Invoice 2. Home screen	No	0	Yes	-	-	25	-	-

Table 1-25 SNAP Case Study

Part 2 – Examples

This page intentionally left blank

Examples

Introduction This chapter includes several examples of how non-functional requirements can be assessed using the categories and sub-categories.

Contents This chapter includes the following examples:

Example 1: Change Free Text to Valid Values

Requirement: An international retail store ordering system has several free-text fields to be replaced by lists of valid values to improve accuracy of data entered and reduce order failures due to validation errors. No new functionality is requested.

Address validation will check country, state, county, city, street and house number. Product description is built in hierarchy of product type, manufacturer, model number and color.

Two elementary processes were identified which were impacted by the requirement and need to be enhanced ("Customer orders a product" and "Customer amends an order").

The solution design involves one sub-category ("Data entry validation"). The SCU is the elementary process.

Address validation is done in 6 nesting levels: first country (1), then state (2), then county (3), then city (4), then street (5) and finally the house number (6). Product validation has 4 nesting levels: first type (1), then manufacturer (2), then model number (3) and then color (4).

The "Customer orders products" EP will execute both the address and product hierarchy validations. Since address validation has 6 nesting levels and product validation has 4 nesting levels, the longest chain of validations to be considered for 1.1 is 'Address validation'. Hence the number of nesting levels to be considered for customer orders product is 6, which gives high complexity.

The "Customer amends an order" EP will execute only the product hierarchy validation which has 4 nesting levels. (4 DETs are nested). Hence the nesting level for this EP will be 4, which gives average complexity.

		1.1 Data Entry Validation			
No.	**SCU Description**	**# DETs**	**# Nesting Levels**	**Formula**	**SP=**
1	**Customer orders products**	**10**	**6**	**=4*#DETs**	**40**
2	**Customer amends an order**	**4**	**4**	**=3*#DETs**	**12**

Table 2-1: Example 1, SP Calculation for Data Entry Validations

Total SP for the project = ∑ SP for all SCU of the sub-category = 40+12 = 52 SP

Example 2: Data Security

Requirement: To meet the new security standards of a bank, it is decided to encrypt the data transferred from one system to another as well as the data displayed on user interfaces. Sensitive Personal Information (SPI) data should be encrypted by the application before passing on to the other systems. Any SPI data to be displayed on user interfaces should be masked by * symbol.

The design for the solution required writing a program to implement encryption of SPI data which will be used by any process in application 1.

The solution design involves one sub-category ("1.3 Data Formatting"). The SCU is the elementary process.

Three elementary processes were identified as impacted: "View subscriber details", "View payment history" (both to mask SPI data by *) and "Customer information extract sent to application 2" (process to send 16 bit encrypted data).

To decide the transformation complexity level, masking data by * would qualify as a Low and encryption of data would qualify as high complexity. The following data is considered as SPI :credit card number, blood group, SSN, telephone number, credit history)

This information is used in all 3 processes.

		1.3 Data Formatting			
No.	**SCU Description**	**Transformation Complexity**	**# DETs**	**Formula**	**SP =**
1	View subscriber details	Low	5	2* #DETs	10
2	View payment history	Low	5	2* #DETs	10
3	Customer information extract sent to application 2	High	5	5 *#DETs	25

Table 2-2: Example 2, SP Calculation for Data Formatting

Total SP for the project = ∑ SP for all SCU of the sub-category = 10+10+25 = 45 SP.

Example 3: Internal Data Backup and Data Transformation

Scenario

An application is designed as three tier solution architecture: User Interface, Middle Layer and Backend Server. Backend server layer holds the database, Middle Layer holds the business processes and User Interface is the front end for the users of the application to view and maintain data.

This application need to enhance to advanced technology platform to create a system that will be more intuitive and easier to use. The platform includes hardware and software that effectively manages connectivity, access, and synchronization. This means that User Interface should support for desktop user and remote device (Handset - as with a smart phone) for portability. To reduce the time and effort, remote device users must be able to download the latest work and access the data. This requires replicating the data from server into the handset.

Requirements

1. Data on the user interface should be grouped in order to improve usability
2. Safety and recoverability: It is required to create a back-up to important data

The three layers of the solution are considered as the three different partitions within this application boundary.

The "View Orders" process will be enhanced to do the data grouping in the middle layer partition and then pass on the data to the user interface partition. The enhancement would be done only in the middle Layer. The "View Orders" process takes a total of 20 DETs as a sum of unique input and output fields at the middle layer and it reads/updates a total of 2 ILFs/EIFs. This solution for requirement 1 involves one sub-category (1.4 Internal Data Movements). SCU is the EP within the Middle Layer partition.

The backup would be created a table in the Backend Layer and the backup process would back up the data from "Order" data file in the backend and delete the data after 1 day. This requires creating a new Back up process and creating a new Back up table within the Backend partition.

The "Order" file in the Backend is replicated in the front end for user access. All the DETs in the "Order" file should be considered here. FTRs are the backend file and the front end replicated file.

The "Back up process" will read 20 DETs from the "Order" data file and update the same in the Back up table. The solution for requirement 2 involves 2 sub categories: 1.4 (Internal Data Movements) for the new backup process, and 3.2 (Database Technology) for the new back up table.

		1.4 Internal Data Movements			
No.	**SCU Description**	**Transformation Complexity**	**# DETs**	**Formula**	**SP =**
1	View Orders	Low	20	4*DETs	80
2	Back Up process	Low	20	4*DETs	80

Table 2-3: Example 3, SP Calculation for Internal Data Movements

		3.2 Database Technology			
No.	**SCU Description**	**FTR Complexity**	**# Changes**	**Formula**	**SP =**
1	Back up Table	Low	1	6*#changes	6

Table 2-4: Example 3, SP Calculation for Database Technology

Total SP for the project = ∑ SP for all SCU of the sub-category = 80+80+6 = 166 SP

Example 4: Creating New Products and Offers

Scenario A telecom software application is designed for easy maintainability and fast launch of new products and offers for its customers. The service provider only needs to create a set of configurations (reference data) to launch a new product without any code change; No logic changes are required for processing the new orders or offers.

Requirements Ten new products are to be launched along with three new offers (An offer is a bundle of products with specific prices. An offer is limited in sale period, it can be offered for a limited period)

Attributes	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	Product 7	Product 8	Product 9	Product 10
LOB	IPTV	IPTV	Internet access	Internet access	Internet access	Land line	Land line	Land line	Mobile	Mobile
Bandwidth	5	10	5-100	20	5	(-)	(-)	(-)	(-)	(-)
# of channels	100	150	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
SLA	Gold/ Silver/ Platinum	Gold/ Silver/ Platinum	Gold/ Silver/ Platinum	Gold/ Silver/ Platinum	Gold/ Silver/ Platinum	Regular /Senior	Regular /Senior	Regular /Senior	(-)	(-)
# of devices	1-5	1-5	1-5	1-5	1-5	(-)	(-)	(-)	(-)	(-)

Table 2-5 Example 4, Main description of the new products

The offers' attributes

1. List of products.
2. Price plan scheme.
3. Discount.
4. Valid from.
5. Valid until.

The design for solution required configuring 10 new products in the "Products" logical file and 3 new offers are to be created in the "Available Offers" logical file. The solution involves only one sub-category, 1.5 (Delivering Added Value to Users by Data Configuration).

Four elementary processes are impacted. "Create Order" and "Modify Order" are the two processes which would consume the product and offer configurations while creating or modifying an order. The "Products" file consists of 10 records to be configured and as there are 10 products, 10 records will be configured. For "Available Offers", 5 attributes to be configured for each offer and as there are 3 offers to be created, 3 records will be configured.

SCU: "Create Order"

		1.5 Delivering Added value to users by Data Configuration			
No.	**The Logical File**	**Complexity Level**	**# Attributes**	**Formula**	**SP=**
1	Products	Low (10 records)	5	6*attributes	30
2	Offers	Low (3 records)	5	6*attributes	30

Table 2-6: Example 4, SP Calculation for Delivering Added Value to Users by Data Configuration, first SCU

SCU: "Modify Order"

		1.5 Delivering Added value to users by Data Configuration			
No.	**The Logical File**	**Complexity Level**	**# Attributes**	**Formula**	**SP=**
1	Products	Low (10 records)	5	6*Attributes	30
2	Offers	Low (3 records)	5	6*attributes	30

Table 2-7 Example 4, SP Calculation for Delivering Added Value to Users by Data Configuration, second SCU

Total SP for the project = ∑ SP for all SCU of the sub-category = 30+30+30+30 = 120 SP

Example 5: Compliance with Standard

Requirements

Compliance with ADA standard 508 or W3C Web Content Accessibility Guidelines (WCAG) 2.0 for accessibility.

Add accessibility options, so that people with difficulties to hear sounds, and people with difficulties to see a normal display, can use the application easily.

The proposed design is:

- Add pop-up icons whenever a sound is generated (there are four different sounds)
- Add big and simple fonts, with one size (14 pt.) to all Menus and Fields on all screens
- One specific color instead of the normal font options.

SNAP Counting process:

- Changing the font size from 10 pt. to 14 pt., changing font colors are considered as technical changes. The requirement is not considered as adding new functionality or changing the functionality, therefore no FP are generated.
- We assume that the icons do not need any computations (no animation).
- The design involves one SNAP sub-category, ("User Interfaces").
- The SCU is the elementary process.
- The new icons and font change affect 5 elementary processes (that do not overlap).
- In two elementary processes, the change is counted as 'Simple'; In 2 Processes, the change is 'Average' and one process is considered Complex.
  (Assuming < 10 GUI properties added/changed for Simple , 10 - 15 properties added/changed for Average and more than 15 properties added/changed for Complex).
- There is a fixed set of UI element for each EP.

Note: The four icons that accompany the sounds are one unique UI element (8 properties: name, type, resolution, size, orientation, open width, location (x), location (y). Fonts appear in the following unique UI elements: menus, icons, up to 11 types of controls, tabs.

- EP 1 – 5 unique UI elements impacted.
- EP 2 – 10 unique UI elements impacted.
- EP 3 – 5 unique UI elements impacted.
- EP 4 – 13 unique UI elements impacted.
- EP 5 – 7 unique UI elements impacted.

		2.1 User Interfaces			
No.	**SCU Description**	**Complexity Level**	**# of Unique UI elements**	**Formula**	**SP =**
1	EP 1	Low	5	2*# unique UI elements	10
2	EP 2	Low	10	2*# unique UI elements	20
3	EP 3	Average	5	3*# unique UI elements	15
4	EP 4	Average	13	3*# unique UI elements	39
5	EP 5	High	7	4*# unique UI elements	28

Table 2-8: Example 5, SP calculations for User Interfaces

Total SP for the project = ∑ SP for all SCU of the sub-category = 10+20+15+39+28 = 112 SP

Example 6: Help

Requirements: Enhancing additional "Help" to an application.

The proposed design:

d) Pop-up screens will appear when the user right-clicks on a field, with an explanation of how and when this field should be used (estimated number of fields: 60, plus 40 context-sensitive).

e) Explanation will include a hyper-link to either a video with a demonstration, or a wizard with a set of Q&A (50 links to Q&A and 15 videos).

The design involves the sub-category Help Methods. The following table shows how help items are sized:

		2.2 Help Methods			
No.	**Help Item**	**Help Type**	**# added Help Items**	**Formula**	**SP=**
1	Pop-up boxes	On-Line Text	60	=2*(#help items)	120
2	Pop-up boxes	Context + On-line	40	=3*(#help items)	120
3	Hyperlinks to Q&A	Context	50	=2*(#help items)	100
4	Hyperlinks to Videos	Context	15	=2*(#help items)	30

Table 2-9: Example 6, SP calculation for Help methods

Total SP for the project = ∑ SP for all SCU of the sub-category = 120+120+100+30 = 370 SP.

Example 7: Adding Input and Output Methods

(Single Instance approach)

Scenario A banking software application supports five different processes (in FP terms elementary processes): Create Account, Modify Account, Make Payment, End-of-Day (EOD) Account Creation summary report, EOD Credit Debit report

Requirements At present, the three elementary processes of 'Create Account', 'Modify Account' and' Make Payment', take input by keying in data from the keyboard. The bank wants to enhance the software to be able to accept input for these three processes in the form of scanned documents and by reading a barcode as well.

(The "Create Account" and "Modify Account" processes 20 DETs each, and "Make Payment" process processes 15 DETs).

The "EOD Account Creation Summary Report" and "EOD Credit Debit Report" are currently sent out in printed CSV format. The bank wants to enhance the software to be able to produce the output for these processes in the form of printed PDF as well as inline mail to the recipients.

(The EOD Account Creation Summary Report has 15 DETs and EOD Credit Debit Report has 10 DETs).

The design solution for this requirement involves two subcategories 2.3 (Multiple Input Methods) and 2.4 (Multiple Output Methods).

		2.3 Multiple Input Methods			
No.	**SCU Description**	**Complexity Level**	**# Additional Input Methods**	**Formula**	**SP =**
1	Create Account	High	2	6* # Additional Input Methods	12
2	Modify Account	High	2	6* # Additional Input Methods	12
3	Make Payment	Average	2	4* # Additional Input Methods	8

Table 2-10: Example 7, SP calculation for Multiple Input methods

		2.4 Multiple Output Methods			
No.	**SCU Description**	**Complexity Level**	**# Additional Output Methods**	**Formula**	**SP =**
4	EOD Account Creation Summary Report	Average	2	4* # Additional Output Methods	8
5	EOD Credit Debit Report	Average	2	4* # Additional Output Methods	8

Table 2-11: Example 7, SP calculations for Multiple Output methods

Total SP for the project =
$\sum$ SP for sub-category 2.3 + $\sum$ SP for sub-category 2.4= 12+12+8+8+8=48

Example 8: Multiple Software Platforms

Requirements While deciding the solution for a software project it was decided that part of it would need to be developed in a multi-platform environment.

The system consists of three tiers. The front-end tier is developed using Java; the middle tier is developed using C++ and the third tier uses SQL.

To ensure that the application can become compatible with different devices, some of the front-end functionality needs to be developed also on a different platform, using XML.

Three elementary processes (EP1, EP2, and EP3) in the application are using all three tiers. (Each is developed as a combination of Java, C++ and SQL)

Two additional processes (EP4, EP5) are doing front-end and middle processing and they do not use the new XML part. (they are developed with a combination of Java and C++)

The solution involves sub-category 3.1 Multiple platforms, but not for all involved EPs: When the architectural framework itself consists of different platforms to deliver part of the functionality, then sub-category 3.1 should not be used. Therefore, EP4 and EP5 are not qualified to have SNAP size using this sub-category.

Moreover, the need for developing the software in multiple software platforms is a technical requirement; hence, it is not covered in FP.

This solution involves subcategory 3.1 Multiple Platforms.

		3.1 Multiple Platforms		
No.	**SCU Description**	**# of Software platforms**	**Same software family?**	**SP =**
1	EP1	2 (XML, JAVA)	No	40
2	EP2	2 (XML, JAVA)	No	40
3	EP3	2 (XML, JAVA)	No	40
4	EP4	1	N/A	None
5	EP5	1	N/A	None

Table 2-12: Example 8, SP calculation for Multiple Platforms

Total SP for the project = ∑ SP for all SCU of the sub-category = 40+40+40= 120 SP.

Example 9: Performance Improvement

Requirements The customer of Telecommunication software applications requires improving the performance of some functionality.

The throughput time to create order and create subscriber need to be improved from average of 2 minutes to 1.5 minutes or less. "View payments" should be improved from 10 seconds to 8 seconds or less for all customers. "Make Payment" transactions need to be improved from 3 seconds to 2 seconds

The design of the solution required the following changes:

1. Tuning SQL queries in "Create Order", "Create Subscriber" and "Make Payment", to make database updates faster. (Using parameterized SQL, improving database connection handling, wise implementation of DB commits etc.)
2. Creation of an indexed view on "Payments" database file, so that "View Payments" can read the view instead of reading the database. (It will improve the performance of "View Payments" process). Corresponding changes in the SQL queries need to be made to read from the new view.
3. The "Create Order" and "Create Subscriber" processes read/update Customer, Subscriber and Order database files. Customer and Subscriber database files have more than 5 RETs and more than 50 DETs, therefore are High complex. The "Order database" file is Average complexity (~30 DETs).
4. "Make Payment" reads/updates Subscriber and Payments database files. Payments database file is an Average complexity FTR; Subscriber FTR is High, therefore the FTR complexity is "High."
5. View Payments reads Payments database file only.

The solution for this requirement involves one sub-category: 3.2 Database Technology.

		3.2 Database Technology					
No.	**SCU Description**	**Highest complexity level of FTRs involved**			**# of database changes**	**Formula**	**SP =**
		# of DETs	**# of RETs**	**Complexity**			
1	Create Order	50	5	High	1	12*# of Changes	12
2	Create Subscriber	40	8	High	1	12*# of Changes	12
3	Make Payment	50	5	High	1	12*# of Changes	12
4	View Payment	25	5	Average	2	9*# of Changes	18

Table 2-13: Example 9, SP calculation for Database Technology

Total SP for the project = ∑ SP for all SCU of the sub-category = 12+12+12+18 = 54 SP

Example 10: Batch Jobs

Requirements

A banking software application provides functionalities to accept all deposits and payments and functionalities to withdraw cash, transfer money and make payments.

The Bank has a requirement to create an EOD (End-of-Day) batch job, which will:

1. Read the following logical database files: Account, Payments and Credits;
2. Apply the said business logic for required calculations and data transformations; and
3. Update two logical database files which are: Credit Summary and Debit Summary.

The batch job is triggered at midnight and processes all the data of the past 24 hours. This batch job doesn't take any input business data and doesn't give out any output business data. The entire scope of the job is limited within the application boundary.

The batch job should read 3 DB files and update 2 DB files. The job processes a total of 25 DETs.

Since there is no input or output crossing the boundary, implementing FP might not be possible. Hence, this requirement involves 1 sub-category: 3.3 Batch Processes.

		3.3 Batch Jobs				
No.	**SCU Description**	**# of FTRs**	**Complexity Level**	**# of DETs**	**Formula**	**SP=**
1	Credit Debit Daily Summary Batch Job	5	Average	25	6*# of DETS	150

Table 2-14: Example 10, SP calculation for Batch Processes

Total SP for the project = 6*25 = 150 SP

Example 11: Using Components

Requirements

A retail customer approaches a software vendor to create a new retail shopping website designed for new target buyers – teen agers (In addition to the web site designed for the traditional consumers). After analyzing the requirements of the customer the vendor lists down different components already developed by the customer for other web sites, and can be used in the current application. The components required are:

1) Login.
2) Display inventory.
3) Compare products.
4) Add and store in shopping cart.
5) Capture customer details.
6) Capture shipping details.
7) Make payment.

The analysis by the vendor team concludes that they can reuse components 1, 4 and 7 from their in-house product offerings without any customization. The remaining functionalities will need to be newly developed / tailored as per the needs of the customer.

The project re-uses 3 in house components.

Assuming one elementary process is involved, the SNAP assessment for component based software development is:

Type	SP Calculation
In-house components	SP=3*3 = 9 SP
Third party components	N/A

Table 2-15: Example 11, SP calculation for Component Based Software

Total SP = ∑ SP for the in-house component = 9 SP

Example 12: Multiple Interfaces

Requirements

A Telecom Rating software application currently receives input from one Network application and sends output to a Billing application.

After acquiring another company (with different applications), the Telecom company has decided to merge the Rating activities into its Rating application.

Calls and Data Usage information should flow to the Rating application from additional two inputs. The Rating application should send all voice and data usage information to additional output.

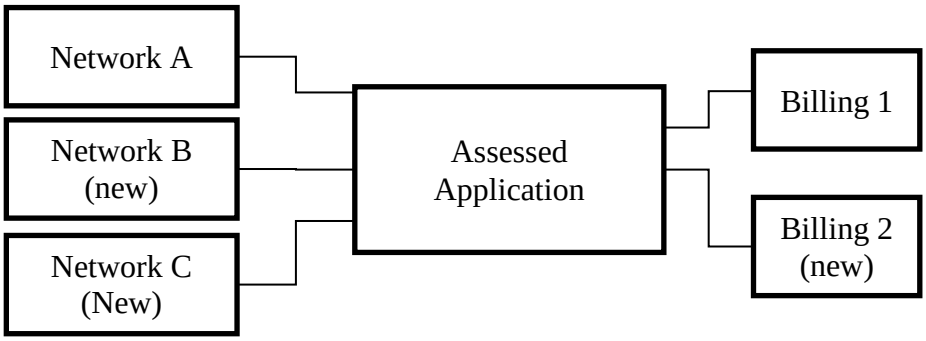


Figure 2-1: Example 12, Required Architecture View

At present, the "Voice usage extract" receives input from External Application and sends the output to Billing. After the change is implemented, it needs to verify the input interface and - based on the input interface - send the output to the corresponding Billing system, after implementing any interface-specific formatting or sorting of the data. The functionality would remain the same and there is no functional enhancement required.

Assuming the Voice Usage Extract and Data Usage Extract takes 20 DETs as input and output.

The design solution involves one sub-category: 4.2 Multiple Input / Output Interfaces

We need to calculate the SP for Input and Output interfaces separately.

Input Interfaces:

No.	SCU Description	4.2 Multiple Input /Output Interfaces			
		Complexity level based on # DETs	**# of additional input Interfaces**	**Formula**	**SP =**
1	Voice Usage Extract	High	2	6 * (Additional # of Interfaces)	12
2	Data Usage Extract	High	2	6 * (Additional # of Interfaces)	12

Table 2-16: Example 12, SP calculation for Multiple Input Interfaces

Output Interfaces:

No.	SCU Description	4.2 Multiple Input/ Output Interfaces			
		Complexity level based on # DETs	**# of additional output Interfaces**	**Formula**	**SP =**
1	Voice Usage Extract	High	1	6 * (Additional # of Interfaces)	6
2	Data Usage Extract	High	1	6 * (Additional # of Interfaces)	6

Table 2-17: Example 12, SP calculation for Multiple Output Interfaces

Total SP for the project = ∑ SP for the inputs and outputs = 12+12+6+6 = 36 SP

Part 3 – Appendices

This page intentionally left blank

Part 3 - Appendices

Introduction

Part 3 provides Appendices on several related topics.

Appendix A provides a Glossary of terms used within the SNAP process and the APM.

Appendix B provides examples of usage and linking between functional and non-functional sizes.

Appendix C provides a document Index.

Contents

Part 3 includes the following sections:

Part 3 - Appendices

This page intentionally left blank

Glossary

Term	Description
APM	Assessment Practice Manual
ASPA	The application SNAP Points after the enhancement project
Assessment Categories	The framework on which the SNAP assessment is based
Assessment Criteria	Information used to determine the values used to develop the assessment
Assessment Questions	Questions that are related to specific attribute(s) which allows for the non-functional assessment of a given sub-category
Assessment Ratings	The answer to an assessment question using given criteria.
Assessment Scope	Defines the set of non-functional user requirements to be included in the assessment
Assessment Value ("SNAP Points")	Non-functional size measure (SP)
Category	A group of components, processes or activities that are used in order to meet the non-functional requirement.
CPM	Counting Practices Manual
Decision Effective Date	Effective date of decisions to update the APM
DET	Data Element Type. Unique, user recognizable, non-repeated attribute
DSP	Development Project SNAP points
EP	Elementary Process. The smallest unit of activity that is meaningful to the user(s)
ESP	Enhanced Project SNAP points

FUR	Functional User Requirements. A sub-set of the user requirements (UR); requirements that describe what the software shall do, in terms of tasks and services. (ISO 14143-1:2007)
IFPUG	International Function Point Users Group
Impact Study	A study that is initiated if there is any possibility that a common practice or several organizations or types of applications may change
NFR	Non-functional User Requirements. 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 user requirement
NFSSC	Non-functional Sizing Standards Committee
NFSSC Review	NFSSC reviews and discusses the rationale for each proposed update and its potential impact which will lead the committee to accept or reject the proposed update
Non-functional Assessment	Size of software in SNAP points
Non-functional Assessment process	Process described it the APM to arrive at a SNAP count
Partition	A set of software functions within an application boundary that share homogeneous assessment criteria and values.
RET	Record Element Type User recognizable sub-group of data element types within a data function
ROI	Return on investment =([Gain from investment] – [Cost of investment]) divided by [Cost of investment])
SCU	SNAP Counting Unit The component or activity, in which complexity and size is assessed. The SCU can be a component, a process or an activity identified according to the nature of one or more sub-categories. In some cases, the SCU is identical to the elementary process.
SNAP	Software Non-functional Assessment Process
SP	SNAP Points
Sub-Category	A component, a process or an activity executed within the project, to meet the non-functional requirement

User Requirements	Requirements describing what the user is asking for. (UR)
User View	A user view is the functional and the non-functional user requirements as perceived by the user

IFPUG APM Link to IFPUG CPM

Introduction

This section describes how the Software Non-functional Assessment Process links to the Function Point Analysis process.

Caution: This is a preliminary view of the linkage between SP and FP. Further analysis is required to determine how the two size measures can be used together. Future releases will further address this issue.

Contents

This appendix provides the SNAP Process diagram and includes examples of potential SNAP uses.

FPA and SNAP Link

Non-functional size can be used in conjunction with functional size to provide an overall view of the project or application including both functional and non-functional sizing.

Assessing the effort impact on projects as a result of the SP is out of scope of this document. Organizations should collect and analyze their own data to determine non-functional productivity impacts.

Potential uses of non-functional size together with functional size are provided by way of example.

Diagram of the Link between FPA and SNAP Processes

The following diagram illustrates the link between the FPA process and the SNAP process.

The Purpose, Scope, and Logical Application boundaries need to be consistent between the FPA and SNAP Processes.

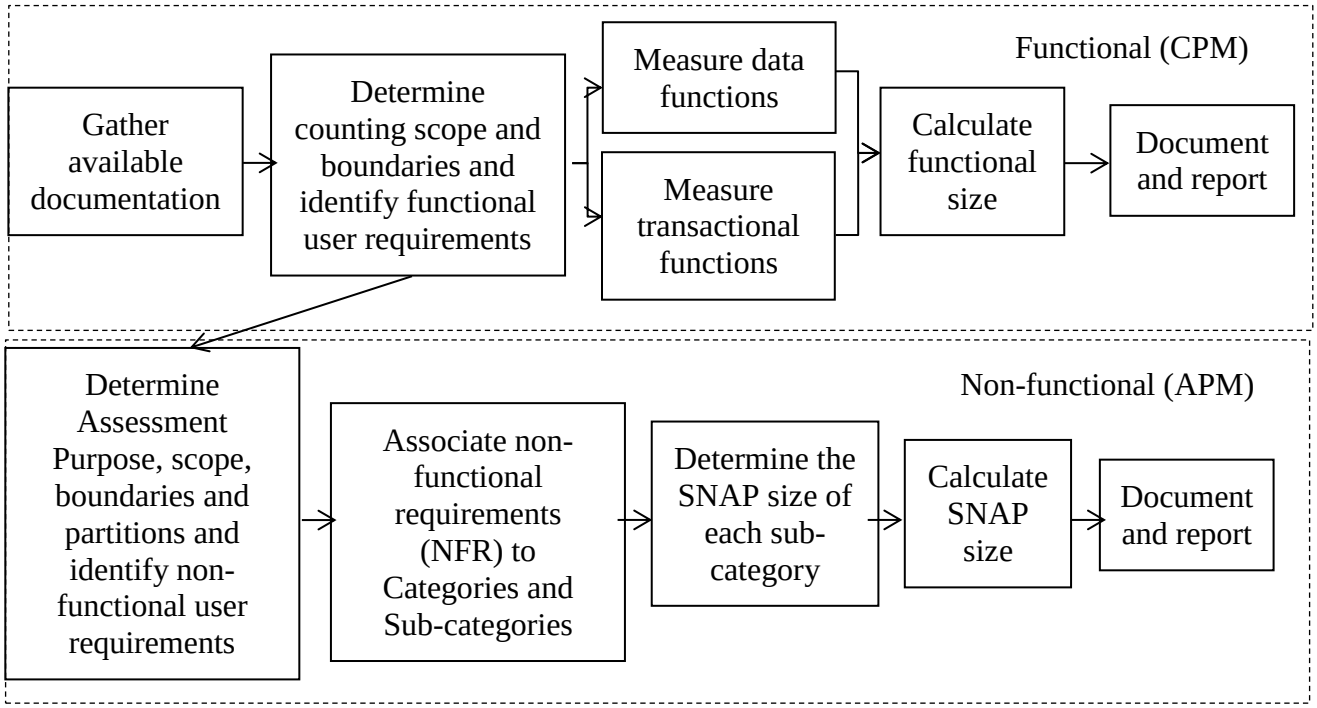


Figure 3-1: The link between FPA and SNAP Processes

Counting Function Points and SNAP Points

A requirement may contain both functional and non-functional aspects. In such a case, the requirement will have a functional size, measured in Function Points, and a SNAP size, measured in SNAP Points.

Such a requirement should be broken down into its functional components and non-functional components, and the segregation should be agreed by both the user/customer and development teams. Use FP for FUR parts of the requirements and SP for the non-functional parts of the requirements.

The following table is a guideline. To define NFR, ISO standard or a similar standard may be used.

Case #	Circumstance	Description	Guideline
1	Requirements are Functional Only	The users do not have any explicit or implicit non-functional requirements	Count function points only
2	Requirements are clearly marked as NFR	Parties agree on clear segregation between functional requirements and non-functional requirements; Requirements classified as NFR cannot be sized with function points	Count SNAP points only
3	Requirements involve both Functional and non-functional aspects	Functional requirements have additional NFR which can be clearly identified:	See table below
4	Requirements are functional only, transactions cross Partitions	Functional requirements may involve single or multiple flows. In case of multiple flows, and using the present CPM guidelines, each flow might not qualify as a separate elementary process.	Count function points to size the new/enhanced functionality for the main elementary process as per CPM, add SNAP size for the transactions / flows within the application's boundary, that cross the partitions
5	Requirements are functional, but they are provided without any software change	Functionality (or any business value) that is added or modified by changing reference data or other means that cannot be sized by function points, according to the present CPM guidelines or FP counting practices of the organization.	Count SNAP points using sub-category 1.5 – Delivering Functionality by Data configuration

Table 3-1 FP and SP interrelations

Requirements Involve Functional and Non-functional Requirement

The following guideline should be used to determine how FP and SNAP points should be counted

The NFRs are associated with the following sub-category:	What to check	Counting Method
1.1 Data Entry Validation	The elementary process	FPA and SNAP. See notes 1 and 2 below
1.2 Logical and Mathematical Operations	The elementary process	FPA and SNAP. See notes 1 and 2 below
1.3 Data Formatting	The elementary process	FPA and SNAP (SNAP sizes changes in the transactions due to non-functional requirements.)
1.4 Internal Data Movements	The elementary process	FPA and SNAP (SNAP sizes data movements between partitions, both for functional and non-functional requirements). See note 3 below
1.5 Delivering Added Value to Users by Data Configuration	The elementary process	FPA or SNAP but not both – see note 4 below
2.1 User Interfaces	Set of screens as defined by the elementary process	FPA and SNAP (SNAP sizes only data elements that does not have a functional change and therefore are not sized by FP) Creation of a new UI element that is FP counted will not generate SP. Modifying a UI element so that functionality is added and therefore FP are counted, will not generate SP. Modifying a UI element with no change in functionality will generate SP.
2.2 Help Methods	The assessed application	SNAP should be counted for the types of "help" that are not FPA counted. Help techniques such as tool tips, dynamic help on mouse over (context help) are SNAP counted. Static web pages are SNAP counted

The NFRs are associated with the following sub-category:	What to check	Counting Method
2.3 Multiple Input Methods	The elementary process	When FP count is used for one input, SNAP should be used for the additional inputs only.
2.4 Multiple Output Methods	The elementary process	When FP count is used for one output, SNAP should be used for the additional outputs only
3.1 Multiple Platforms	The elementary process	When FP count is used for one platform, SNAP should be used for the additional platforms only
3.2 Database Technology	The elementary process	Table changes, views, database partitions, database capacity change and changes in queries or inserts that do not change the functionality: SP only. When adding tables or columns for non-functional purpose only, SNAP should be used
3.3 Batch Processes	The batch jobs	When a batch file is not qualified as any of the transaction function per FPA guidelines, use SNAP to size the batch jobs. When an existing batch job is divided into multiple jobs and the reason is to meet non-functional requirements, use SNAP to size the batch jobs
4.1 Component Based Software	The elementary process	FPA and SNAP
4.2 Multiple Input / Output Interfaces	The elementary process	FPA when creating the functionality SNAP when adding interfaces without modifying the functionality FPA and SNAP when modifying functionality and adding interfaces

Table 3-2 FP and SP interrelations rules

Notes:

1. SNAP introduces an additional non-functional size to functionality that has been sized before, using function points only. Therefore, the overall size of the application includes now

a separation between the functional aspects of the non-functional aspects.

2. A requirement must be separated into its functional aspects and its non-functional aspects. The functional aspects are sized using function points and the non-functional aspects are sized using SNAP.

   As an example, a new or enhanced functionality may be delivered with validation, while the validation itself has no impact on the functional size.

   SNAP adds a non-functional size when validation is used. As a result, the overall size (function points and SNAP points) increases.

3. Partition:
   Two scenarios can appear here:
   1. An elementary process (which delivers functionality on its own without invoking several other flows or processes) crosses multiple partitions.
   2. An elementary process that consists of multiple flows, which perform several tasks. These flows might perform different tasks of fetching, transforming, formatting and processing data, and creating outputs which are finally consumed by the elementary process. These flows are not independent and user identifiable as per CPM guideline, and cannot be broken-up into several elementary processes.

4. Delivering Added Value to Users by Data Configuration

   When functionality is added using a software change and configuration, and it generates function points, do not add SNAP size. SNAP size is to be used only when no software code or database structure changes are involved

Potential Uses of SNAP Points

Project Estimation

Project Estimation

- Count FPs for project – utilize productivity rates based on FP size and project type to estimate functional project effort etc., from data contained in repositories.
- Complete SNAP assessment – utilize historical data to determine impact of SNAP score on project effort. Adjust project effort up or down depending on SNAP score to provide a functional/technical effort estimate.
  The relationship between SNAP and effort may be specific to an organization.
  Once significant SNAP assessment data is collected, a "rule of thumb" relationship may be established by industry, platform, etc, from data contained in repositories.
- Complete risk/attribute Assessment – Assesses organizational factors that impact productivity. Utilize historical data on risk/attribute impacts on effort and adjust project effort up or down appropriately to provide a functional/technical/risk effort estimate.

Note: Can use all three effort estimates individually or in any combination to produce an estimate range. The productivity ratio for FP should reflect only the effort required to implement the functional requirements and the SNAP productivity ratio should reflect only the effort required to implement the non-functional requirements.

ROI

Calculate Return on Investment (ROI) Estimate of Application Replacement

- Count FPs for Application to determine functional size of the application
- Complete SNAP assessment to determine the non-functional size of the application

Compare cost of the replacement project and the future maintenance costs with the maintenance cost of the existing application to determine the ROI.

Emphasis Area(s)

Utilize SP to indicate the non-functional areas of emphasis in the project/application

- Examine the detail responses to the SNAP assessment compared to ISO/IEC 25010 characteristics (or another standard for classifying non-functional requirements) to ensure the appropriate focus

Example: a project with more SP allocated to accuracy versus attractiveness indicates that there is more emphasis on accuracy.

Options Comparison

Comparison of non-functional alternatives by having an overview of total costs (functional and non-functional)

Examples:

- Using COTS versus development
- Compare technologies that deliver functionality in order to select the appropriate technology (e.g., database technologies, interface technologies)

Maintenance Cost

Assist in overall assessment of maintenance costs/resources

- Use FP to size the functionality of the application
- Use SP to size the non-functional characteristics of the application

Use historical data to estimate the maintenance costs by functional and non-functional size for each application to plan future resources and aid in maintenance strategies and annual budgeting.

Non-functional Measure (SNAP Points)

The IFPUG CPM has been transformed into an ISO standard for functional size measurement with the exclusion of the General System Characteristics (GSCs), which assess the general functionality of the application. (IFPUG FPA V4.1 unadjusted was received as ISO standard from 2003 - ISO/IEC 20926:2003 - later in 2009 for v4.3). The most important consideration in relation to these issues is how the non-functional requirements affect the size. Since they are not functional requirements, they do not contribute to the functional size. However, they are still part of the overall requirements (functional and non-functional) for the software, and therefore contribute to the overall size of requirements.

SP have been developed to be used independently of the GSCs to assess the non-functional requirements. Practitioners of the SNAP Point assessment should NOT use both the SP and GSCs simultaneously. Using both may inflate the resulting non-functional size of the non-functional requirements. SP can be used in conjunction with function points (given identical boundaries) to provide an overall view of a project or application including both functional and non-functional sizing.

Index

