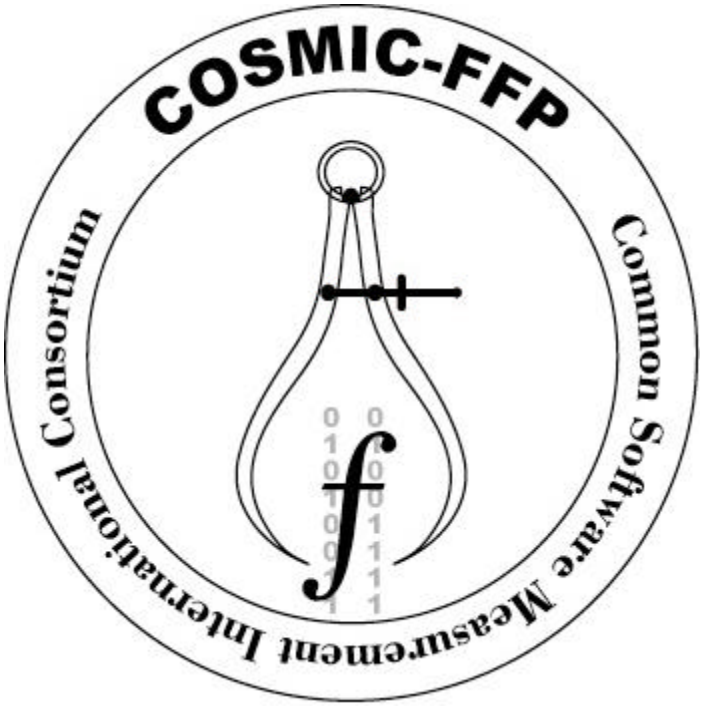


MANUAL DE MEDICIÓN

VERSIÓN 2.1

Mayo 2001

Traducción al español

Reconocimientos

AUTORES: (en orden alfabético)

- Alain Abran, UQAM – Software Engineering Management Research Laboratory,
- Jean-Marc Desharnais, Software Engineering Laboratory in Applied Metrics,
- Serge Oligny, Bell Canada,
- Denis St-Pierre, DSA Consulting Inc., Canadá,
- Charles Symons, Software Measurement Services Ltd., Reino Unido

TRADUCTORES:

Francés:

- Jean-Marc Desharnais, Software Engineering Laboratory in Applied Metrics, Montreal, Canadá
- Émile Sayag, Alcatel, France

Español:

- Luis Molinié, Laboratoire de recherche en gestion des logiciels, Montreal, Canada
- Mariana Miranda, Montreal, Canada

Italiano:

- Roberto Meli, Data Processing Organization eri, Roma, Italia

Japonés:

- Moritsugu Araki, JECS System Research Co. Ltd., Gunma, Japón

Portugués:

Si usted tiene comentarios sobre la traducción española, por favor comuníquese con Luis Molinié, (molinie.luis@uqam.ca)

REVISORES DE LA VERSIÓN 2.1 (en orden alfabético):

Alain Abran, Université du Québec à Montréal – UQAM, Canadá	Pam Morris, Total Metrics, Australia*
Moritsugu Araki, JECS Systems Research, Japón	Risto Nevalainen, Software Technology Transfer Finland OY, Finlandia *
Guinter Guerhen, Büren & Partner, Alemania	Marie O'Neill, Software Measurement Services – SMS, Irlanda
J. M. Desharnais, Software Engineering Lab in Applied Metrics – SELAM, Canadá	Serge Oligny, Bell Canada
Reiner Dumke, University of Magdeburg, Alemania	Jolijn Onvlee, The Netherlands *
Peter Fagg, Software Measurement Services – SMS, Reino Unido	Grant Rule, Software Measurement Services Ltd., Reino Unido
Vinh T. Ho, Institut Francophone d'Informatique – IFI, Vietnam	Denis St-Pierre, DSA Consulting Inc., Canadá
Roberto Meli, Data Processing Organization eri, Italia	Charles Symons, Software Measurement Services Ltd., Reino Unido

REVIORES DE LA VERSIÓN 2.0

- Moritsugu Araki, JECS Systems Research, Japón
- Fred Bootsma, Nortel, Canadá
- Denis Bourdeau, Bell Canada, Canadá
- Pierre Bourque, UQAM, Canadá
- Guinter Guerhen, Büren & Partner, Alemania
- Sylvain Clermont, Hydro Québec, Canadá
- David Déry, CGI, Canadá
- Gilles Desoblins, Francia
- Martin D'Souza, Total Metrics, Australia
- Reiner Dumke, University of Magdeburg, Alemania
- Peter Fagg, Reino Unido *
- Thomas Fetcke, Alemania
- Eric Foltin, University of Magdeburg, Alemania
- Anna Franco, CRSSM, Canadá
- Paul Goodman, Software Measurement Services Ltd., Reino Unido
- Nihal Kececi, University of Maryland, Estados Unidos
- Robyn Lawrie, Australia
- Ghislain Lévesque, UQAM, Canadá
- Roberto Meli, Data Processing Organization eri, Italia
- Pam Morris, Total Metrics, Australia*
- Risto Nevalainen, Software Technology Transfer Finland OY, Finlandia *
- Jin Ng, Hmaster, Australia
- Patrice Nolin, Hydro Québec, Canadá
- Marie O'Neill, Irlanda
- Jolijn Onvlee, The Netherlands *
- Geert Poels, Catholic University of Louvain, Belgica
- Laura Primera, UQAM, Canadá
- Paul Radford, Charismatek, Australia
- Eberhard Rudolph, Alemania
- Grant Rule, Software Measurement Services Ltd., Reino Unido*
- Richard Stutzke, Science Applications Int'l Corporation, Estados Unidos
- Ilionar Silva, UQAM, Canadá
- Vinh T. Ho, UQAM, Vietnam

* Miembros fundadores del grupo COSMIC y autores de COSMIC-FFP

[1] Una versión de dominio público del Manual de Medición COSMIC-FFP y de otros informes técnicos se hallan disponibles en las páginas Web http://www.cosmicon.com y http://www.lrgl.uqam.ca/ffp.html

Control de versión

El cuadro siguiente resume los cambios de la versión inglesa.

FECHA	REVISOR(ES)	Modificaciones
99-03-31	Serge Oligny	Primera versión, para comentarios de los revisores
99-07-31	Ver reconocimientos	Revisada, incluyendo los comentarios de los revisores
99-10-29	Miembros del equipo COSMIC	Revisada con los comentarios finales antes de la publicación de la versión “pruebas de campo”
01-05-01	Miembros del equipo COSMIC	Revisado de conformidad con ISO/IEC 14143-1: 1998 + clarificaciones sobre las regla de medición

Prefacio

La versión del método de medición COSMIC-FFP es el resultado del feedback recibido sobre el uso en la práctica de la versión original de este manual durante las pruebas de campo realizadas en al año 2000.

Todos los comentarios recibidos fueron analizados, consolidados y formalmente circulados entre los miembros del equipo COSMIC. Como resultado, no se encontró necesario hacer cambios a los principios de base COSMIC-FFP. Por lo tanto, los cambios en el texto de esta última versión son ante todo de naturaleza editorial, para clarificar la comprensión, y en algunos casos, para mejorar las definiciones de ciertos conceptos COSMIC-FFP de base. Dichos cambios son mostrados en el anexo E.

El método COSMIC-FFP fue aceptado por ISO/IEC JTC1-SC7 en el año 2000 como un nuevo item de trabajo para la standardización. La secciones siguientes de este documento fueron remitidos a ISO en mayo del 2001 para su revisión de standardización : El glosario y los capítulos 1 al 5.

Dos secciones de este documento no han sido todavía completadas : las secciones 6 (convertibilidad) y 7 (COSMIC-FFP medidos antes del desarrollo del software). El contenido de esas secciones será adicionado posteriormente.

Los miembros del equipo COSMIC
Mayo, 2001.

Contenido

Glossario

Los términos siguientes serán usados en el presente manual de medición de acuerdo a las definiciones de esta sección. Para los términos que hayan sido definidos por ISO, como "Medición de Talla Funcional" o "Escala", la definición ISO será adoptada.

Para algunos términos, el sufijo "tipo" es adicionado entre paréntesis. Esto es para enfatizar que en casi todo el manual esos términos significan "tipos" y no "ocurrencias". Sin embargo, en el texto principal el sufijo "tipo" será omitido a fín de facilitar la lectura, excepto cuando sea específicamente necesario de distinguir entre tipo y ocurrencia.

Acoplamiento (Coupling): El acoplamiento es una medida de la fuerza de la interconexión entre procesos funcionales. El acoplamiento depende de la complejidad de la interfase entre los procesos funcionales, del punto de entrada o de referencia a un proceso funcional, y del tipo de información que pasa a través la interfase. El acoplamiento es medido según una escala ordinal (acoplamiento alto o bajo).

Ambiente operativo del software: es el conjunto de sistemas software que son ejecutados de manera concurrente en un sistema informático específico.

Atributo (de datos) (-tipo) (sinónimo de "tipo elemental de información): Un atributo de datos es la más pequeña parcela de información, dentro de un grupo definido de datos, que tiene significado desde el punto de vista de los requerimientos funcionales del usuario del software.

Cantidad de base[2]: Es una cantidad que en un sistema cuantitativo es aceptada, por convención, como funcionalmente independiente de otra cantidad.

Capa: Una capa es el resultado de una partición funcional del ambiente del software en el cual todos los procesos funcionales incluidos poseen el mismo nivel de abstracción.

En un ambiente multi-capa, una capa intercambia datos con otra capa a través sus respectivos procesos funcionales. Esas interacciones son de naturaleza jerárquica. Si las capas son consideradas por parejas, una capa es el "cliente" de otra capa. Una capa "cliente" utiliza los servicios funcionales provistos por otra capa subordinada. El método de medición define intercambios de mismo nivel (“peer-to-peer”) como el intercambio de datos entre dos ítems de software en la misma capa.

Componente funcional de base (CFB): Un componente funcional de base es una unidad elemental de los requerimientos funcionales del usuario, definidos por un método de medición de talla funcional para propósito de medición.[3].

Entrada (-tipo): Una ENTRADA (E) es un movimiento de atributos de información, que se encuentra en un grupo de datos en el lado del usuario de la frontera del software, hacia el interior de dicha frontera. Una ENTRADA (E) no actualiza los datos (la información) que ella desplaza. Funcionalmente, un sub-proceso de entrada traslada los datos del lado del usuario de la frontera y los pone a disposición del proceso funcional al cual él pertenece. Hay que tener en cuenta que en el COSMIC-FFP, una entrada incluye también algunos sub-procesos asociados a la manipulación de datos (validación) – Para más detalles, ver 4.1.

Escala de referencia: Para cantidades particulares de una categoría determinada, una escala de referencia es un conjunto ordenado de valores, continuos o discretos, definidos por convención como referencia para clasificar las cantidades según su magnitud[4].

---

[2] De "International Vocabulary of Basic and General Terms in Metrology”, International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.

[3] De “ISO/IEC 14143-1998 – Software Engineering – Software measurement – Functional size measurement – Part 1: Definition of concepts”, clause 3.1.

[4] De "International Vocabulary of Basic and General Terms in Metrology”, International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.

Escritura (-tipo): Una ESCRITURA (W por WRITE) refiere los atributos de datos de un grupo de datos. Funcionalmente, un sub-proceso de ESCRITURA envía datos desde el interior del proceso funcional, al cual el pertenece, hacia el exterior de la frontera, al almacenamiento de datos. Hay que tomar en cuenta que en el COSMIC-FFP, una escritura incluye algunos sub-procesos asociados a la manipulación de datos (validación) – Para más detalles, ver 4.1.

Evento (-tipo): Ver "evento desencadenador".

Evento desencadenador (-tipo): Un evento desencadenador (*Triggering event*) ocurre al exterior de la frontera del software medido e inicia uno o más procesos funcionales. Los relojes y los eventos temporales pueden ser eventos desencadenadores. Dado que cada capa identificada es separada por una frontera, los eventos desencadenadores pueden ocurrir en una capa e iniciar procesos funcionales en otra.

Frontera: La frontera de un software es la línea conceptual que separa ese software y el ambiente en el cual él opera, de la manera percibida por sus usuarios desde un punto de vista externo. La frontera permite al medidor (la persona que mide) de distinguir , sin ambigüedad, lo que está incluido dentro del software medido de lo que hace parte del ambiente externo dentro del cual el software funciona.

Función de medición (COSMIC-FFP): Es la función matemática que asigna un valor numérico a una variable sobre la base del patrón de medida COSMIC-FFP. La variable de la función de medición COSMIC-FFP es el sub-proceso.

Grupo de datos (-tipo): Un grupo de datos es un conjunto no nulo, no ordenado y no redundante de atributos, en el cual cada uno de los atributos describe un aspecto complementario del mismo objeto de interés (ver definición). Un grupo de datos es caracterizado por su perspectiva (ver definición).

Input: Datos para los cuales los valores son independientes del software y que son utilizados por éste durante su funcionamiento. La definición genérica utilizada en este manual es notablemente diferente de la definición específica usada por el International Function Point Users Group (IFPUG) para Input. Para COSMIC FFP, un input contiene todas las entradas que un proceso funcional particular implica.

Lectura: Una LECTURA (R por READ) refiere a los atributos de un grupo de datos. Funcionalmente, un sub-proceso de LECTURA toma los datos almacenados y los pone a disposición del proceso funcional al cual ella pertenece. Hay que tomar en cuenta que en el COSMIC-FFP, una LECTURA incluye algunos sub-procesos asociados a la manipulación de datos (validación) – Para más detalles, ver 4.1.

Medición de talla funcional: Es el proceso por el cual la talla funcional de un sortware es medida[5].

Método de medición[6]: Es la secuencia lógica de las operaciones descritas genéricamente y usadas en la ejecución de mediciones.

Método de medición de la talla funcional: Es una implementación específica de una medición de la talla funcional, definida por un conjunto de reglas conforme a lo establecido en las características obligatorias de la parte 1 del documento ISO/IEC 14143-Parte 1[3].

Modelo[7]: Es una descripción o analogía usada para ayudar a visualizar un concepto que no puede ser observado directamente.

Movimiento de datos (-tipo): Un movimiento de datos es definido como un sub-proceso, identificado en el software medido, el cual mueve uno o más atributos de información que pertenecen a un sólo grupo de datos. Existen cuatro tipos de movimientos de datos: entrada, salida, lectura y escritura.

Objeto de interés (-tipo): Un objeto de interés es identificado desde el punto de vista de los requerimientos funcionales del usuario, y puede ser un objeto físico, un objeto conceptual o una parte de un objeto conceptual en el mundo del usuario. El objeto de interés es requerido para el procesamiento y/o el almacenaje (conservación) de datos (información).

---

5 De "ISO/IEC 14143-1:1998 – Software Engineering – Software measurement – Functional size measurement – Part I: Definition of concepts", clause 3.7.

6 De "International Vocabulary of Basic and General Terms in Metrology", International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.

7 Adaptado de Merriam Webster's Collegiate Dictionary, 10th Edition.

Output: Datos cuyo valor depende de la ejecución del software y que son creados o modificados por el software durante su ejecución (Operación). La definición genérica de output, usada en este manual, es notablemente diferente de la definición específica usada por el International Function Point Users Group (IFPUG). Para COSMIC FFP, un output contiene todas las salidas que un proceso funcional particular implica.

Patrón de medida (COSMIC_FFP): 1 Cfsu (Unidad e talla funcional Cosmic o *Cosmic Funcional Size Unit*) es definido como un movimiento elemental de datos.

Persistencia (de un grupo de datos o información): Es la calidad que indica cuan largo es el período durante el cual un grupo de datos es retenido en el contexto de los requerimientos funcionales del usuario. Tres categorías de persistencia son definidas: transitoria (mientras dura del proceso funcional), corta (más allá de la duración del proceso funcional pero mientras el software está en operación), e indefinida (más allá de la duración de la operación del software).

Procedimiento de medición[8]: Es el conjunto de operaciones descritas específicamente y usadas en la implementación de una medición particular, de acuerdo a un método dado.

Proceso funcional (-tipo) (Sinónimo de "Tipo de transacción): Un proceso funcional es un conjunto único de movimientos de datos (entrada, salida, lectura, escritura) que implementa un conjunto coherente y logicamente indivisible de requerimientos funcionales del usuario. Un proceso funcional es desencadenado directa, o indirectamente via un "actor", por un Evento (-tipo), y es completado cuando se ejecuta todo lo que es requerido de ser hecho en respuesta a evento desencadenador (-tipo).

Requerimientos funcionales del usuario (FUR por *Funcional User Requirements***):** Es una expresión ISO que designa un subconjunto de requerimientos del usuario, sobre la base de la perspectiva del usuario. El FUR representa las prácticas y procedimientos de el usuario que deben ser realizados por el software para responder a las necesidades del primero. Los FURs excluyen los requerimientos de calidad y otros requerimientos técnicos[9].

Salida (-tipo): Una SALIDA (X) es un movimiento de atributos de información que se encuentra en un grupo de datos al interior de la frontera del software hacia el lado del usuario de dicha frontera. Una SALIDA no lee los datos que ella desplaza. Funcionalmente, un sub-porceso de SALIDA envía los datos desde el proceso funcional al cual ella pertenece (implícitamente al interior de la frontera del software) y los pone a disposición del lado del usuario de la frontera. Hay que tomar nota que en el COSMIC-FFP, una SALIDA incluye también algunos sub-procesos asociados a la manipulación de datos (validación) –Para más detalles, ver 4.1.

Software[10]: Es un conjunto de instrucciones, datos procedimientos y eventualmente de documentos que operan como un todo para responder a un conjunto de objetivos específicos, los cuales pueden ser descritos desde una perspectiva funcional, como un conjunto finito de requerimientos funcionales, técnicos y de calidad del usuario.

Sub-proceso (–tipo) (COSMIC FFP): Un sub-proceso COSMIC-FFP es un movimiento de datos que ocurre durante la ejecución de un proceso funcional. Existen cuatro tipos de movimientos de datos: entrada, salida, lectura y escritura, cada uno de los cuales incluye un sub-proceso específico de manipulación de datos – para más detalles, ver 4.1. Un sub-proceso COSMIC- FFP equivale a la un tipo componente de base funcional. Un sub-proceso COSMIC-FFP expresa sólamente los requerimientos funcionales del usuario y excluye los requerimientos técnicos y de calidad.

Tipo de componente funcional de base : una categoría definida de componente funcional de base[3]. El método de medición COSMIC-FFP ofrece cuatro tipos de componentes funcionales de base : los tipos de Entrada, Salida, Lectura y Escritura.

---

[8] De "International Vocabulary of Basic and General Terms in Metrology", International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.

[9] De "ISO/IEC 14143-1998 – Software Engineering – Software measurement – Functional size measurement – Part 1: Definition of concepts", clause 3.8.

[10] Adaptado de Merriam Webster's Collegiate Dictionary, 10th Edition, and La Petit Larousse Illustré, 1996 Edition.

Unidad de medida[11]: Es una cantidad particular, definida y adoptada por convención, a la cual son comparadas otras cantidades de la misma categoría, a fín de expresar sus magnitudes con relación a ella. Es de notar que las unidades de medida les son asignados símbolos y nombres, por convención.

Usuarios: Seres humanos o dispositivos software o de ingeniería que interactúan con el software medido.

Valor (de una cantidad)[12]: Es la magnitud de una cantidad particular, generalmente expresada como una unidad de medida multiplicada por un número.

[11] De "International Vocabulary of Basic and General Terms in Metrology", International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.

[12] De "International Vocabulary of Basic and General Terms in Metrology", International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.

1

INTRODUCCIÓN

El software es un componente muy importante en muchos presupuestos corporativos. Las organizaciones reconocen la importancia de controlar los gastos en software y de analizar la performance de los montos asignados al desarrollo y al mantenimiento de software, a fin de compararse con las mejores prácticas. Por ese motivo, las medidas y los modelos que usan esas medidas son necesarios.

Para analizar la calidad y la productividad asociadas al desarrollo y al mantenimiento de software, la utilización de medidas es necesaria. Por un lado, las medidas técnicas son necesarias para cuantificar la performance técnica de los productos y servicios desde un punto de vista informático. Las medidas técnicas pueden ser usadas para analizar la eficiencia, para mejorar la performance de los diseños, etc.

Por otro lado, las medidas funcionales son necesarias para cuantificar la performance de los productos o servicios desde el punto de vista del usuario o del propietario, para analizar la productividad, entre otras cosas. Las medidas funcionales deben ser independientes de las técnicas de desarrollo y de las decisiones de implementación. Ellas pueden así ser utilizadas para comparar la productividad de diferentes técnicas y tecnologías.

Los puntos de función tradicionales[13] son un ejemplo de medidas funcionales. Ellos son disponibles para el campo de los software de gestión (MIS), en el cual han sido usados extensivamente para el análisis de productividad y estimaciones (Abran, 1996; Desharnais, 1988; Jones, 1996; Kemerer, 1987). Esta medida captura con éxito las características funcionales específicas del software MIS.

Sin embargo, los puntos de función tradicionales han sido criticados por no ser aplicables a todos los tipos de software. (Conte, 1986; Galea, 1995; Grady, 1992; Hetzel, 1993; Ince, 1991; Jones, 1988; Jones, 1991; Kan, 1993; Whitmire, 1992). D.C. Ince describe el análisis de los puntos de función tradicionales de la siguiente manera:

> *"A problem with the function point approach is that it assumes a limited band of application types: typically, large file-based systems produced by agencies such as banks, building societies and retail organizations, and is unable to cope with hybrid systems such as the stock control system with a heavy communication component."* (Ince, 1991, page 283)

Es reconocido que los puntos de función tradicionales no permiten capturar todas las características del software en tiempo real. Cuando ellos son utilizados para medir ese tipo de software, se obtiene un valor que no constituye una medición adecuada de su talla funcional. Así, hasta la llegada de la versión 1.0 de los FFP, no existía una medida de talla funcional que considerara procedimientos detallados para la medición del software en tiempo real.

[13] Albrecht A.J., Gaffney Jr. J.E., "Software function, source lines of code and development effort prediction: a software science validation", IEEE Transactions on Software Engineering, Vol. SE-9, pp. 639-648, November 1983.

La versión 1.0 de los FFP fue propuesta, en setiembre de 1997[14], con el objeto de ofrecer una medida de talla funcional específicamente adaptada al software de tiempo real. Poco después, la utilización práctica de los FFP en varias organizaciones y las pruebas de campo, realizados por el Software Engineering Management Research Laboratory de la UQAM y el Software Engineering Laboratory in Applied Metrics, demostraron que los FFP no solamente permiten de capturar la talla funcional de los software en tiempo real, sino que también pueden capturar la talla de los software técnicos y los sistemas de explotación. Además, la pruebas de campo han demostrado que los FFP son aplicables también a la medición de la talla funcional de los software MIS[15].

Los resultados de las pruebas de campo y el *fedback* recibido de las organizaciones que han utilizado la versión 1.0 de los FFP han motivado a los autores a mejorar el método. Muchas de la mejoras propuestas en este manual fueron inspiradas en el trabajo del COSMIC group[16], el cual propone también los principios de una nueva generación de métodos de medición de talla funcional. Los resultados de esos esfuerzos están contenidos en este documento que constituye la versión 2.0 del método de medición COSMIC-FFP.

La versión 2.0 del método de medición COSMIC-FFP ha sido también concebida para asegurar su entera conformidad con la norma ISO 14143-1 y los principios del grupo COSMIC.

ACERCA DE LA INICIATIVA COSMIC

Dado el explosivo crecimiento y la diversidad de los contratos de outsourcing en el campo del software, los proveedores y los clientes de esos servicios se han visto en la necesidad de contar con instrumentos más precisos de estimación y de medición de la performance, los cuales deben ser igualmente fiables para todos los tipos de software. Los actuales métodos de medición de la talla de software no poseen siempre las características que el mercado necesita, o no son adaptados que para determinados tipos de software. La industria necesita urgentemente medidas de talla funcional que deben ser, al mismo tiempo, más precisas y más ampliamente utilizables.

La iniciativa COSMIC apunta a satisfacer esas necesidades de: a) los proveedores de software que deben traducir los requerimientos del cliente en la talla del software como un paso clave en la estimación de los costos del proyecto y, b) los clientes que quieren conocer la talla funcional del software recibido como un componente importante de la medición de la performance del proveedor.

---

[14] St-Pierre D., Maya M., Abran A., Desharnais J.-M., Bourque P., "Full Function Points: Counting Practices Manual", Technical Report 1997-04, Université du Québec à Montréal, Montréal, Canada. Disponible en el Web al URL: www.lrgl.uqam.ca/ffp.html

[15] Oligny, S.; Abran, A.; Desharnais, J.-M.; Morris, P. , *Functional Size of Real-Time Software: Overview of Field Tests*, in Proceedings of the13th International Forum on COCOMO and Software Cost Modeling, Los Angeles, CA, October 1998.

[16] Para mayor información, ver www.cosmicon.com.

2

RESEÑA DEL MÉTODO DE MEDICIÓN COSMIC-FFP[17]

El método de medición COSMIC-FFP es una medida normalizada de software. Esta sección presenta y discute el modelo del proceso de medición general COSMIC-FFP y explica sus componentes claves: el modelo COSMIC-FFP de talla funcional y los principios subyacentes a las fases de establecimiento de la correspondencia o representación y de medición cuando este modelo es usado para la medición de software. En esta sección se discute también el contexto de utilización de la medición de la talla funcional.

APLICABILIDAD DEL MÉTODO COSMIC-FFP

El método de medición COSMIC-FFP está diseñado para su aplicación al software en los siguientes dominios:

- Software de aplicación, que típicamente soporta la administración de negocios bancarios y de seguros, la contabilidad, y la gestión de personal, de compras, de la distribución y de la producción, entre otros. Este tipo de software es caracterizado por ser intensivo en datos y su complejidad es dominada ampliamente por la necesidad de manejar grandes cantidades de datos sobre los eventos del mundo real.
- El software en tiempo real, cuya tarea es de tener bajo control los eventos del mundo real. Algunos ejemplos de este tipo de software serían los sistemas para intercambios de comunicaciones telefónicas y de mensajes, los software incorporados en los dispositivos de control de los motores que hacen funcionar los artefactos domésticos, los ascensores o los automóviles sobre la base del control y la adquisición automática de datos, y los software que constituyen los sistemas operativos de las computadoras.
- Los software híbridos de los dos anteriores, como los sistemas de reservación en tiempo real de las líneas aéreas u hoteles.

El método de medición COSMIC-FFP (versión 2.0) no ha sido diseñado para medir la talla funcional de software, o parte de ellos, que:

- son caracterizados por algoritmos matemáticos complejos u otras reglas especializadas y complejas, como las los sistemas expertos, de simulación, los software de aprendizaje, los sistemas de proyección meteorológica, etc.
- procesan variables continuas como los sonidos audio o las imágenes de video que se encuentran, por ejemplo, en los juegos de computadora y en los instrumentos de música.

Para dichos softwares es posible de definir extensiones locales del método de medición COSMIC-FFP. La sección 4.1.5 de este manual explica en qué contextos esas extensiones locales podrían ser utilizadas y provee un ejemplo de una extensión local. Cuando son utilizadas, esas extensiones locales deben ser tratadas según las convenciones presentadas en la sección 5 de este manual.

2.1 MODELO DEL PROCESO DE MEDICIÓN COSMIC-FFP

El método de medición COSMIC-FFP implica la aplicación de reglas y procedimientos a un software dado,

[17] Los términos software, input, output, atributos de información (datos) y usuarios son usados en esta sección del manual según las definiciones del glosario.

como éste es percibido desde la perspectiva de los requerimientos funcionales del usuario (FUR). El resultado de la aplicación de esas reglas y procedimientos es un número "valor de una cantidad"[18] que representa la talla funcional del software, desde la perspectiva del usuario. Esas reglas y procedimientos están descritas en las secciones 3 y 4 de este manual.

El método de medición está diseñado para ser independiente de las decisiones de implantación del software a ser medido en los artefactos operacionales. Para alcanzar esta característica, la medición es aplicada a un modelo genérico de los requerimientos funcionales del software (FUR), modelo que corresponde a los artefactos del software a medir. La figura 2.1.1 representa este proceso:

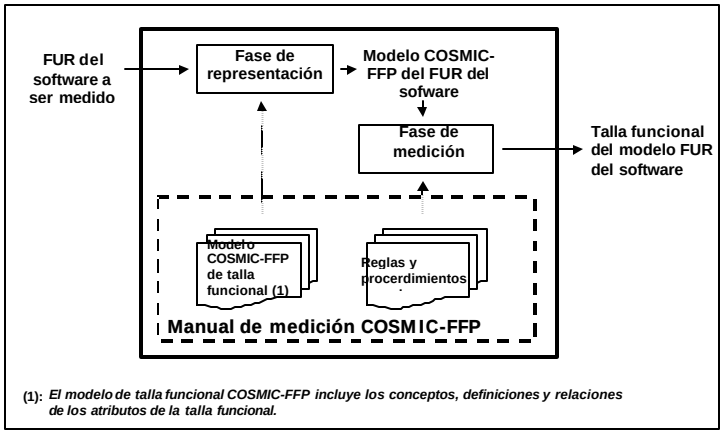


Figura 2.1.1 – Modelo del proceso de medición COSMIC-FFP

El modelo del proceso de medición COSMIC-FFP ilustra que, previamente a la aplicación de las reglas y procedimientos de medición, los FUR del software deben ser representados ("mapeados") en un modelo COSMIC-FFP específico que capture los conceptos, definiciones y relaciones (estructura funcional) requeridos por el ejercicio de medición de la talla funcional. Los detalles de las definiciones, principios y reglas necesarios a la correspondencia del software con el modelo de software COSMIC-FFP son presentados en la sección 3 de este manual.

Una vez que los FUR del software han sido correspondidos o representados dentro del modelo FUR COSMIC-FFP del software, la medición es realizada aplicando al modelo un conjunto de principios y reglas. Dicha aplicación producirá un valor numérico de una cantidad que representa la talla funcional del modelo FUR del software. Por convención, ese valor numérico la talla funcional del software. Los detalles de las definiciones, principios y reglas para la medición del modelo de software COSMIC-FFP son presentados en la sección 4 de este manual.

2.2 EXTRACCIÓN DE LOS REQUERIMIENTOS FUNCIONALES DEL USUARIO

Un sistema software puede ser analizado desde varios puntos de vista. Desde la perspectiva del método de medida COSMIC-FFP, el interés es las funcionalidades que él provee al usuario. Esas funcionalidades son descritas en los FUR. En la práctica, los FUR existen algunas veces bajo la forma de un documento específico (especificaciones requeridas, por ejemplo), pero muchas veces los FUR tienen que ser derivados de otros artefactos u objetos de ingeniería de software. Como es ilustrado en la figura 2.2.1, los FUR pueden ser derivados de objetos de ingeniería de software que son producidos antes que el software mismo exista (típicamente a partir de los documentos de concepción y arquitectura). De esta manera, la talla funcional del software puede ser medida antes de su implementación.

[18] Tal cual que definido por ISO, ver el glosario.

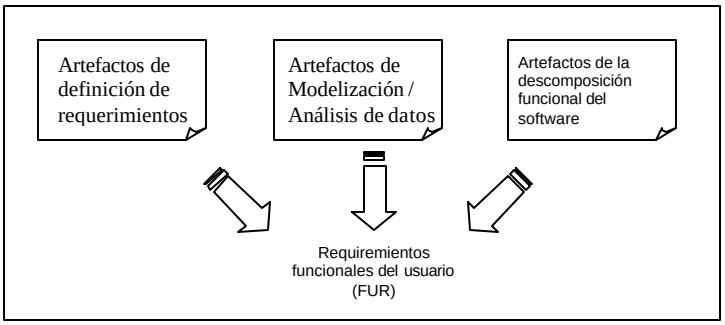


Figura 2.2.1 – Modelo COSMIC-FFP de pre-implementación de los Requerimientos funcionales del usuario

En otras circunstancias, el software puede ser utilizado sin que sean disponibles, ningún, o solamente pocos objetos de arquitectura o de concepción, y que los FURs no sean documentados. En dichas circunstancias, es todavía posible derivar los requerimientos funcionales del usuario de los objetos ya instalados en el sistema informático, tal cual se ilustra en la figura 2.2.2.

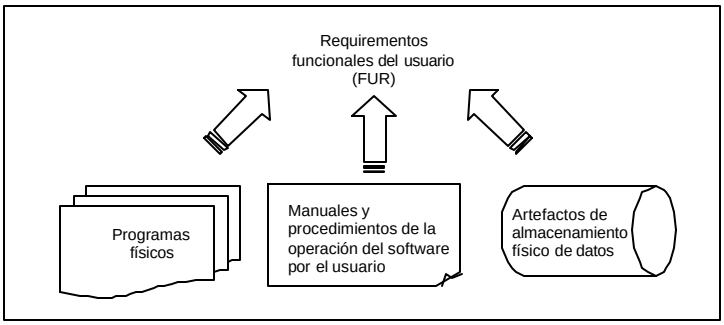


Figure 2.2.2 – Modelo COSMIC-FFP de post-implementación de los Requerimientos funcionales del usuario

Obviamente, el esfuerzo requerido por la extracción de los FURs de diferentes tipos de objetos de ingeniería de software variará. La naturaleza de los FURs es constante, independientemente de los objetos usados en extraerlo. Los Requerimientos funcionales del usuario proveen siempre una descripción de las funcionalidades libradas a los usuarios.

2.3 FASE DE REPRESENTACIÓN COSMIC-FFP

La fase de representación COSMIC-FFP considera como input los requerimientos funcionales del usuario de un software, obtenidos según lo descripo en la sección 2.2, y, usando un conjunto definido de reglas y procedimientos genera un modelo específico de software (el modelo de software COSMIC-FFP), el cual es apropiado para medir la talla funcional. El modelo de software generado corresponde al sub-conjunto de los FUR a ser incluídos en el ámbito específico de la medición de la talla funcional. El modelo de software COSMIC-FFP comprende dos partes: el modelo de contexto y el modelo de software, los cuales son descritos a continuación.

2.3.1 Modelo de contexto COSMIC-FFP

Un aspecto clave de la medición de la talla funcional del software consiste en establecer lo que es considerado como parte del software y lo que es considerado como parte de su ambiente de operación. La figura 2.3.1.1 ilustra el flujo genérico de atributos de datos desde una perspectiva funcional. Según esta perspectiva, se puede observar lo siguiente:

- El software está delimitado por el hardware. En la llamada dirección "front-end", el software está delimitado por componentes del hardware tales como el "mouse", el teclado, la impresora, la pantalla, los sensores o los relais. En lo que está designado como dirección posterior ("back-end"), el software está delimitado por el hardware de almacenamiento como el disco duro y las memorias RAM y ROM.

- El flujo funcional de los atributos de datos puede ser caracterizado por cuatro distintos tipos de movimiento. En la dirección "front-end", dos tipos de movimientos (ENTRADA y SALIDA) permiten el intercambio de atributos de datos con los usuarios. En la dirección "back-end", dos tipos de movimientos (LECTURA y ESCRITURA) permiten el intercambio de atributos de datos con el hardware de almacenamiento.

En tanto que método de medición, el COSMIC-FFP se orienta a medir la talla del software sobre la base de los requerimientos funcionales del usuario. Una vez definidos, esos requerimientos son asignados al hardware y al software según una perspectiva unificada de un sistema informático que intergra esos dos componentes. Pero, dado que el COSMIC-FFP mide la talla del software, solamente los requerimientos que conciernen al software son considerados. (A notar que, en principio, el método COSMIC-FFP puede ser aplicado a los FURs antes que éstos sean aplicados al software o a otra cosa, independientemente de la decisión de eventual de alocación. Esta última aseveración debe, sin embargo, ser verificada en la práctica).

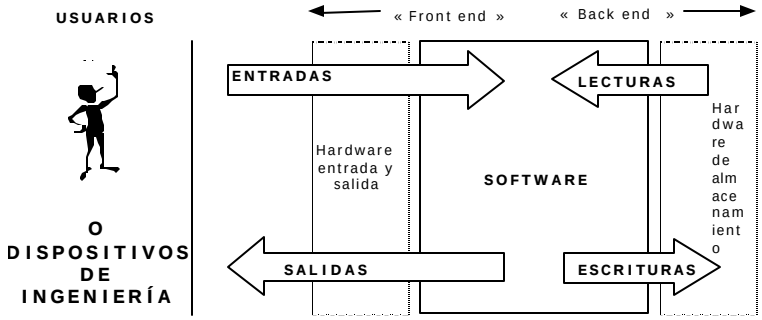


Figura 2.3.1.1 – Flujo genérico de atributos de datos desde una perspectiva funcional

Consideremos ahora dos casos de alocación de requerimientos funcionales de software: una aplicación MIS típica y un software "multi-piezas".

CASO 1 – SOFTWARE MIS TÍPICO

En este ejemplo, , los FURs son asignados al nivel "aplicación" del software ("Aplicación X) en la figura 2.3.1.2). Las otras piezas de software son generalmente requeridas y utilizadas sin modificación alguna.

Desde que: a) todos los requerimientos funcionales del usuario son asignados a una sola pieza de software, y b) esta pieza puede ser fácilmente distinguida de las otras piezas por medio de una interfase bien definida, resulta relativamente simple la identificación de las funcionalidades ofrecidas por la aplicación X, y también, la medición de las funcionalidades al interior de la frontera de esta aplicación. En este ejemplo, y con fines de medición de la talla funcional, se puede hacer una abstracción, ignorando todas las otras piezas de software excepto la aplicación X. En ese caso, la frontera de la aplicación X (línea continua alrededor de la caja gris en la figura 2.3.1.2) no podría ser distinguida de la frontera del medio ambiente del software (línea discontinua que delimita el software del hardware en la figura 2.3.1.2).

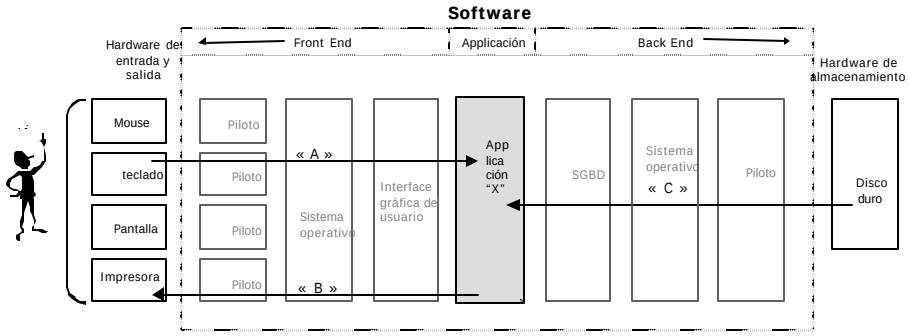


Figura 2.3.1.2 – Ejemplo de asignación de requerimientos para software MIS

CASO 2 – SOFTWARE "MULTI-PIEZAS"

En este ejemplo, se supone que una parte de los requerimientos funcionales del usuario son asignados a una aplicación ("Aplicación A" en la figura 2.3.1.3) y otra parte es asignada a otra pieza de software ("Interfase gráfica del usuario" en la figura 2.3.1.3). Se supone también que, la interfase funcional entre la Aplicación A y la Interfase gráfica del usuario es estructurada y definida en los FURs. Las otras piezas de software son utilizadas sin modificación y el software requerido (compuesto por A y la Interfase gráfica del usuario) simplemente hace uso de dichas piezas.

Existe una relación de dependencia entre la Aplicación A y la Interfase gráfica del usuario: A controla la Interfase gráfica del usuario invocando la funcionalidad provista por la segunda a fin de librar su propia funcionalidad. Como consecuencia de esta relación, la funcionalidad provista por la Interfase no es visible (de manera separada) para el usuario. El usuario ve solamente la funcionalidad que la aplicación A le provee. Para medir la talla funcional del total de requerimientos es necesario distinguir entre esas dos piezas. Por eso, el método de medición COSMIC-FFP considerará a cada una de las piezas en una capa diferente y que cada pieza tendrá sus propias fronteras. En este ejemplo, dado que la interfase entre las dos piezas es estructurada y bien definida, la distinción no crea grandes dificultades prácticas para la medición. Finalmente, hay que destacar que para medir su talla funcional, cada pieza será considerada como un "usuario" de otra pieza.

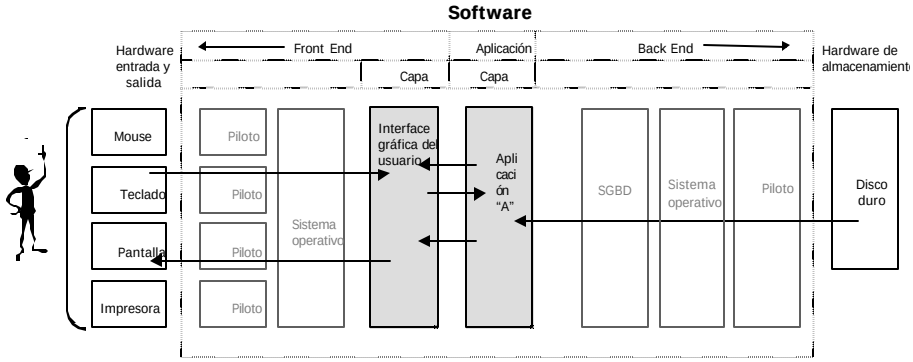


Figure 2.3.1.3 – Ejemplo de asignación de requerimientos para software "multi-piezas"

En la práctica de la ingeniería de software, los requerimientos funcionales del usuario no son necesariamente asignados a una pieza de software. En casos donde una arquitectura de software[19] exista al principio del proceso de desarrollo (como en el caso de la arquitectura ilustrada en la figura 2.3.1.3), la asignación de dichos requerimientos a piezas específicas en esta arquitectura tiene un impacto sobre la talla funcional de cada pieza.

[19] Existen varios modelos de arquitectura en uso. El modelo de capas ha sido usado con fines de ilustración. Otros modelos pueden ser también usados si ellos proveen una perspectiva funcional del software.

En la práctica, para los interesados solamente en la talla global de esos requerimientos, no es necesario considerar la manera como los requerimientos son asignados (Caso 1). Pero, para los interesados en la talla de las porciones de estos requerimientos, el esquema de asignación mostrado en el caso 2 resulta relevante para la identificación de la frontera del software.

El método de medición COSMIC-FFP provee así una herramienta (el concepto de capas de software[20]) para ayudar a diferenciar los requerimientos funcionales del usuario asignados a los diferentes niveles de abstracción funcional. Más formalmente, el modelo de contexto de software COSMIC-FFP ofrece las siguientes características:

Característica 1 –Capas

El software a ser medido puede ser particionado en una o más piezas, de manera a que cada pieza opere a un nivel de abstracción funcional diferente en el medio de operación del software, y que exista una relación jerárquica entre cada nivel particular de abstracción basado sobre los intercambios funcionales entre ellos. Cada nivel es tratado como una capa distinta. Cada capa encapsula las funcionalidades útiles a la capa que utiliza sus servicios en la relación jerárquica y utiliza la funcionalidad provista por la capa "más baja" en esta relación. Hay que destacar que un software dentro de una capa puede ser desarrollado y su talla necesitar ser medida como piezas separadas. Las comunicaciones entre piezas separadas dentro de la misma capa son conocidas como comunicaciones al mismo nivel ("peer-to-peer"). Una discusión más detallada sobre las capas de software es presentada en el Anexo D.

Característica 2 – Frontera

En una capa dada, la pieza de software a ser medida puede ser claramente distinguida de las capas circundantes por una frontera. Una frontera implícita existe así entre cada capa identificada. La frontera es un conjunto de criterios percibidos a través los requerimientos funcionales del usuario del software, que permite establecer una distinción clara entre los ítems que son parte del software (dentro de la frontera) y los ítems que son parte del medio ambiente en el cual el software opera (fuera de la frontera). Por convención, todos los usuarios de una pieza de software están situados al exterior de la frontera de esta pieza de software.

Característica 3 – Usuarios del software

Es posible identificar uno o más usuarios que benefician de las funcionalidades provistas por una pieza de software dentro de una capa. Por definición, los usuarios pueden ser seres humanos, dispositivos de ingeniería u otros systemas software. También por definición, las piezas de software dentro de las capas vecinas inmediatas son consideradas como usuarios cuando interactúan con la pieza medida.

Característica 4 – Requerimientos funcionales del usuario

Las partes de los requerimientos del software que describen la naturaleza de las funciones a ser provistas son designadas "Requerimientos funcionales del usuario" y que son usadas con la única perspectiva de medir la talla funcional del software. Las partes de los requerimientos que describen la manera como las funciones del software tienen que ser implementadas, como la calidad o los requerimientos técnicos, NO son consideradas para medir la talla funcional del software[21].

2.3.2 El modelo de software COSMIC-FFP

El modelo de software COSMIC-FFP asume que los siguientes principios son aplicables al software a ser medido:

---

[20] El término "capa" es usado en este manual según la definición del glosario. Aunque ella contiene un concepto similar al que se observa en arquitectura de software, el uso en este manual no está restringido a esa forma de arquitectura.

[21] Según "ISO/IEC 14143-1998 – Software Engineering – Software measurement – Functional size measurement – Part 1: Definition of concepts".

Principio general 1: *El software a ser representado y medido es alimentado por entradas de datos y genera salidas de datos útiles a los usuarios.*

Principio general 2: *El software a ser representado y medido manipula parcelas de información designadas como grupos de datos, las cuales contienen atributos de datos.*

La Figure 2.3.2.1 ilustra el modelo de software propuesto por el método ce medición COSMIC-FFP. Según este modelo, los requerimientos funcionales de software son implementados por un conjunto de procesos funcionales. Cada uno de esos procesos funcionales es un conjunto único de sub-procesos. Cada sub-proceso realiza, sea un movimiento, sea una manipulación de datos.

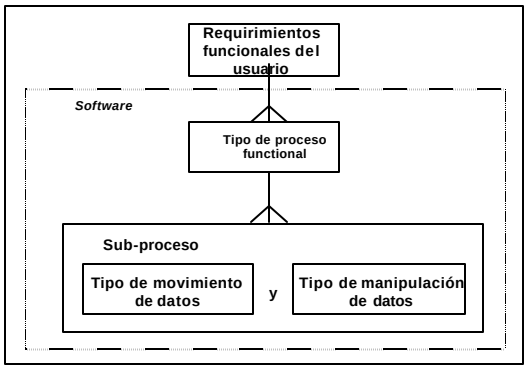


Figura 2.3.2.1 – Un modelo general de software para la medición de la talla funcional

Tal que identificados en el modelo de contexto (Figura 2.3.1.1), el modelo general de software COSMIC-FFP distingue cuatro tipos de sub-procesos de movimiento de datos: entrada, salida, lectura y escritura. Todos los sub-procesos de movimiento de datos transfieren datos contenidos solamente en un grupo de datos. Las entradas transfieren datos desde los usuarios hacia el interior de la frontera del software; las salidas transfieren datos desde el interior de la frontera del software hacia los usuarios; las lecturas y escrituras transfieren datos desde y hacia las formas de almacenaje. Esas relaciones son ilustradas en la Figura 2.3.2.2.

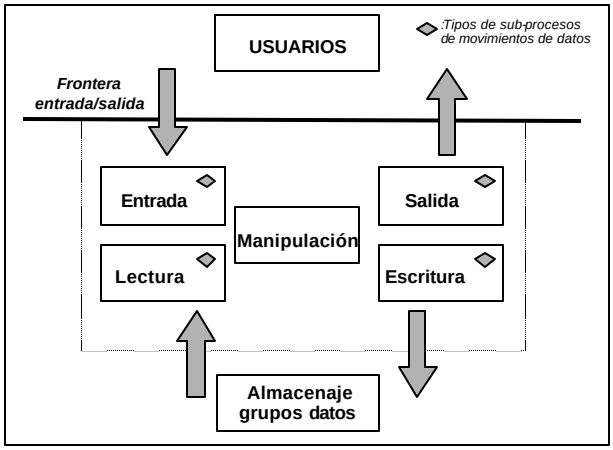


Figura 2.3.2.2 – Los tipos de sub-procesos y algunas de sus relaciones

Como una primera aproximación, los tipos de sub-procesos de manipulación de datos, ilustrados en la Figura 2.3.3.2, son considerados dentro de la definición de movimientos de datos en esta versión del método de medición. Esto es porque los conceptos y definiciones, requeridos para la medición de la manipulación de datos, son todavía objeto de grandes debates y discusiones entre los especialistas en ingeniería de software. El detalle de las funcionalidades incluidas en cada tipo de sub-proceso es descrito en

la sección 4.1 de este manual. Para las organizaciones que deseen tener en cuenta explícitamente los sub-procesos de manipulación de datos, un mecanismo que permite definir extensiones locales al método es provisto en la sección 4.1.5. Para dichos casos, la sección 5 del manual contiene varias convenciones especiales de presentación de la información.

Utilizando los conceptos, definiciones y estructura del método de medición COSMIC-FFP, los requerimientos funcionales del usuario extraídos de los artefactos de una pieza de software son representados en el modelo de software COSMIC-FFP, generando consecuentemente, una instancia de ese modelo. Dicha instancia del modelo contendrá todos los elementos requeridos para la medición de la talla funcional del software, al mismo tiempo que ignoraráa la información no relevante para la medición de la talla funcional.

Las reglas y procedimientos de medición COSMIC-FFP son entonces aplicadas a este modelo COSMIC-FFP instanciado a fin de producir un valor o una cantidad que represente la talla funcional del software. Por lo tanto, dos fases consecutivas son necesarias para medir la talla funcional del software: la representación de los artefactos del software a ser medido en el modelo de software COSMIC-FFP y la medición de los elementos específicos de ese modelo de software.

La fase de representación recibe como input los requerimientos funcionales del usuario extraídos de los artefactos del software a ser medido (tal cual ellos son encontrados o documentados en la organización o tal cual ellos son inferidos del software existente) y genera como output una instancia del modelo de software COSMIC-FFP. Este modelo es instanciado a partir de la utilización de los conceptos y definiciones de la sección 3 y resumidos en los anexos B y C. El modelo de software COSMIC-FFP puede ser presentado en forma de matriz, en la cual las filas describen los procesos funcionales (los que deben ser agrupados en capas), las columnas los grupos de datos y las celdas contienen los sub-procesos identificados. Esta presentación del modelo de software COSMIC-FFP figura en al Anexo A.

2.4 FASE DE MEDICIÓN COSMIC-FFP

La fase de medición COSMIC-FFP considera como entrada una instancia del modelo de software COSMIC-FFP y, haciendo uso de un conjunto definido de reglas y procedimientos, produce un valor de una cantidad cuya magnitud es directamente proporcional a la talla funcional del modelo, sobre la base del principio siguiente:

> **Principio de medición COSMIC-FFP:** La talla funcional del software es directamente proporcional al número de sus sub-prcesos de movimientos de datos.

Por convención, este valor numérico representa la talla funcional del software.

Las características del conjunto de reglas y procedimientos que rigen la generación de ese valor numérico son las siguientes:

Característica 1 – Función de medición

Para cada instancia de un subproceso, es asignada una cantidad numérica, según su tipo, a través de una función de medida.

Característica 2 – Unidad de medida

La norma de medición, 1 Cfsu (Cosmic Functional Size Unit), es definida por convención como equivalente a un movimiento de datos a nivel de sub-proceso.

Característica 3 – Aditividad

La talla funcional de un proceso funcional es definida como la suma aritmética de las tallas funcionales de los sub-procesos que lo conforman. Por extensión, la talla funcional de una pieza de software (en cualquier capa del modelo de software) es la suma aritmética de las tallas de los procesos funcionales contenidos en esa pieza.

La talla funcional de cada uno de los *cambios* en los FURs de una pieza de software es, por convención, la suma aritmética de las tallas funcionales de todas los sub-procesos adicionados, modificados y eliminados en esa pieza de software.

Característica 4 – Sub-unidad de medida

Cuando un grado de precisión adicional sea requerido en la medición de los movimientos de datos, una sub-unidad de medida puede ser definida; por analogía, un metro puede ser sub-dividido en 100 centímetros o 1000 milímetros. Las pruebas sobre el terreno a ser llevadas a cabo permitirán analizar la relevancia de utilizar el movimiento de un sólo dato como una sub-unidad de medida.

Según estas características y utilizando 1 Cfsu como patrón de medida, es interesante de destacar que la talla funcional más pequeña de una pieza de software es de 2 Cfsu, lo que se basa en el principio general 1, que establece que un proceso funcional, por pequeño que sea, tiene menos una Entrada y una Salida o una Escritura. Además, según estas características, no existe un límite superior para la talla funcional de una pieza de software y, sobre todo, no existe límite superior alguno para la talla funcional de ningún proceso funcional.

La talla funcional es así derivada de la aplicación del conjunto documentado de reglas y procedimientos presentado en la sección 4 y resumido en los anexos B y C.

2.5 CONTEXTO DE MEDICIÓN DE LA TALLA FUNCIONAL

Esta sección aborda dos aspectos centrales de la medición de la talla funcional del software: el primero en relación al objetivo de la medición, el segundo referente a la escala de los resultados de la medición.

SOBRE LOS OBJETIVOS DE LA MEDICIÓN

Existen varias razones para medir la talla funcional del software, así como existen varias razones para medir el área de una casa. En un contexto particular, puede ser necesario medir la talla funcional del software antes de su desarrollo, como puede ser necesario medir el área de una casa antes de su construcción. En otro contexto, podría ser útil medir la talla funcional del software *a posteriori*, esto es, después de su puesta en operación, como podría ser útil medir el área de una casa después que ésta es entregada al propietario.

Las razones por las cuales una medida es tomada tienen un impacto, aunque a veces sutil, sobre lo que está siendo medido, el objeto o su representación, sin afectar la unidad de medida o los principios de medición.

En el ejemplo de la casa, la medición del área de su superficie antes de la construcción es basada, obviamente, sobre los planos del edificio. Las dimensiones requeridas (largo y ancho) son extraídas de los planos utilizando escalas apropiadas y el área de la superficie es calculada de acuerdo a una definición normal en geometría para las superficies rectangulares: el producto del largo por el ancho.

Del mismo modo, la medición de la talla funcional del software antes de su desarrollo es basada sobre los "planos" del software, que son en este caso una colección de artefactos producidos antes del desarrollo. las dimensiones requeridas (componentes funcionales de base) son extraídos de esos artefactos utilizando convenciones apropiadas (desde la perspectiva de los requerimientos funcionales del usuario, excluyendo aspectos técnicos y de calidad), y la talla es calculada según una función específica de medición.

Profundizando la analogía de la casa, la medición del área despues de su construcción implica un procedimiento de medición algo diferente, ya que las dimensiones requeridas (largo y ancho) son extraídas del mismo edificio utilizando un instrumento diferente (cinta para medir). Aunque el objeto físico a ser medido difiere (la casa en lugar de los planos), las dimensiones, la unidad de medida (incluida la escala adoptada) y los principios de medición serán los mismos. Alternativamente, la casa puede ser siempre medida sobre la base de los planos, siempre y cuando esos planos sean actualizados.

Asimismo, la medición de la talla funcional del software después su puesta en operación supone la utilización de diferentes procedimientos de medición cuando las dimensiones requeridas son extraídas de

diferentes artefactos. Aunque la naturaleza de esos artefactos difiere, las dimensiones, la unidad de medida y los principios de medición siguen siendo los mismos.

Corresponde al medidor (la persona que mide o la responsable de la medición), basado en el objetivo de la medición, determinar si el objeto a ser medido es la casa verbalmente descrita por su propietario, la casa descrita a través los planos o la casa que ha sido ya construida. El mismo razonamiento se aplica a la medición del software.

Este manual de medición define y describe las dimensiones, unidades de medida y principios de medición para la talla funcional del software. Él provee indicaciones para ayudar a identificar esas dimensiones en algunos de los artefactos generalmente reconocidos. Este manual, sin embargo, no describe ni propone, al medidor, razones específicas para realizar la medición. Sobre la base de los objetivos de la medición, corresponde al medidor determinar los artefactos más apropiados a utilizar para una medición específica.

Sin embargo, los usuarios del método COSMIC FFP son advertidos de la importancia de definir el objetivo y el ámbito de la medición antes de comenzar un ejercicio particular de medición. Por ejemplo, si el objetivo es medir la talla funcional de un software librado por un equipo de proyecto particular, será necesario, primero, de delimitar las fronteras de los diferentes componentes a librar. Esto puede incluir software que ha sido utilizado una sola vez para convertir datos del software que está siendo reemplazado. Si el objetivo es cambiado para medir sólo el nuevo software que será disponible al usuario, la talla será menor, ya que el software usado en la conversión no será incluido en el ámbito de la medición.

SOBRE LAS ESCALAS DE MEDICIÓN

En el ejemplo de la casa, se menciona que la medición de la superficie a partir de los planos implica un factor de escala. Es sabido que los planos son dibujados utilizando escalas. Sea cual fuera la escala escogida, ésta es siempre explicita (i. e. 1:4 o 1:10) y su utilización implica una equivalencia estricta y constante entre las dimensiones medidas en el plano y las dimensiones correspondientes al edificio.

Este factor de escala, constante y bien definido, está ausente de los artefactos producidos por los procesos de ingeniería de software. En este sentido, la ingeniería de software es más próxima de la biología donde las dimensiones son medidas a varios "niveles" (i.e. molecular, célula, tejido, órgano, sistema). Aunque existen algunos factores de escala inherentes a esos "niveles", las relaciones no son constantes.

El método de medición COSMIC-FFP es actualmente documentado al nivel en que todos los grupos de datos son claramente identificados, lo que es ilustrado en la Figura 2.5.1. La utilización del método de medición COSMIC-FFP sobre artefactos de nivel más elevado de abstracción requiere del uso de factores de escala, tal cual son descritos en la sección 7 de este manual. Actualmente, se están realizando trabajos para identificar esos factores de escala. Los resultados de esos trabajos serán presentados en ediciones posteriores a este documento.

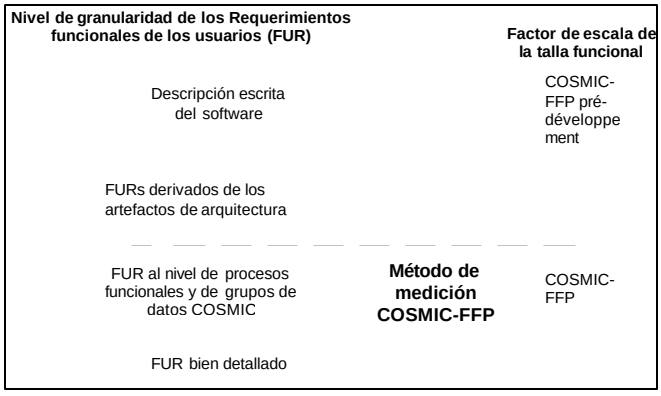


Figura 2.5.1 – Escalas COSMIC-FFP

2.6 OTROS ASPECTOS DE LA TALLA

El método de medición de la talla funcional COSMIC-FFP no presume todos aspectos de la talla del software. Otras dimensiones de la talla del software no son así capturadas por este método. La inclusión de esos aspectos requeriría de acuerdos comunes sobre la definición de "talla" en el caso del software. Dichos acuerdos constituyen, sin embargo, materia de debate y de trabajos posteriores.

3

FASE DE REPRESENTACIÓN – REGLAS Y MÉTODO

El método de medición COSMIC-FFP considera la medición de la talla funcional del software a través dos fases distintas: la representación del software a ser medido en el modelo de software COSMIC-FFP y la medición de los aspectos específicos de este modelo de software.

Esta sección presenta las reglas y el método[22] para el proceso de representación. El método general para representar el software en el modelo genérico de software COSMIC-FFP es resumido en la Figura 3.1.

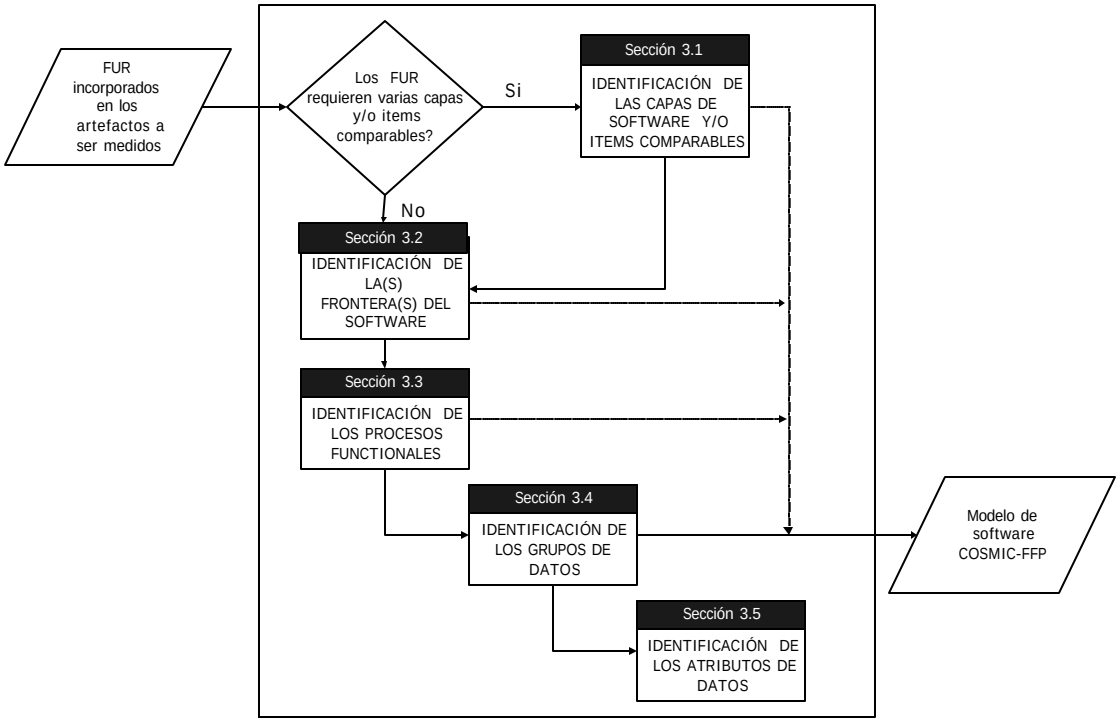


Figura 3.1 – Método general para el proceso de representación COSMIC-FFP

Cada etapa de este método es el objeto de una sub-sección específica (indicada en la Figura 3.1) cuyas definiciones y reglas son presentadas acompañadas de principios de aplicación y ejemplos.

El método general esbozado en la Figura 3.1 ha sido construido para su aplicación en un abanico bastante amplio de artefactos software.

Un procedimiento más sistemático y detallado ofrecería reglas de correspondencia para un conjunto más amplio de artefactos específicos, permitiendo de disminuir la ambigüedad a través la generación del modelo de software COSMIC-FFP. Dicho procedimiento sería, por definición, altamente dependiente de la naturaleza de los artefactos, los cuales, a su vez, dependen le la metodología de ingeniería de software utilizada en

[22] El término “método” es utilizado tal cual es definido por ISO. Para más detalles, ver el Glosario.

cada organización. Las versiones futuras del método de medición COSMIC-FFP incluirían procedimientos de representación para las metodologías más conocidas y ampliamente utilizadas.

3.1 IDENTIFICACIÓN DE LAS CAPAS DE SOFTWARE

Los requerimientos funcionales del usuario pueden explicitar, implicar o el analista de medición puede inferir, que los FURs se aplican al software en diferentes capas o items comparables, para los que la talla debe ser medida separadamente. Alternativamente, el analista de medición puede encontrarse con que el software aparece en diferentes capas o items comparables. En ambos casos, se necesitará de directivas que ayuden a decidir si los FURs o el software comprende una o más capas o items comparables. Las capas pueden ser identificadas según la definición y principios siguientes[23]:

DEFINICIÓN – Capa

Una capa es el resultado de una partición del software en la cual todos los procesos funcionales son ejecutados a mismo nivel de abstracción.

En un ambiente de software de capas múltiples, el software de una capa intecambia datos con el software de otra capa a través sus procesos funcionales respectivos. Dichas interacciones son de naturaleza jerárquica; cuando ellas son consideradas por pares, una capa es "cliente" de la otra. Una capa "cliente" utiliza los servicios funcionales provistos por otras capas subordinadas. El método de medición define al mismo nivel (peer-to-peer) los intercambios entre dos ítems de software que intercambian datos en la misma capa.

La identificación de capas constituye un proceso iterativo que es afinado con el avance de la puesta en correspondencia (representación, modelización). Una vez identificadas, cada capa candidata debe cumplir con los siguientes principios:

PRINCIPIOS – Capa

3. El software en todas las capas provée funcionalidades a los usuarios (Un usuario puede ser un ser humano, un dispositivo físico u otro software).
4. El software en una capa subordinada provee servicios funcionales al software en las capas clientes.
5. El software en las capas subordinadas puede ejecutarse sin la asistencia del software en una capa cliente.
6. El software en una capa cliente puede no ejecutarse apropiadamente si el software en la capa subordinada que depende no es ejecutado apropiadamente.
7. El software en la capa cliente no usa necesariamente todas las funcionalidades provistas por el software en una capa subordinada.
8. El software en una capa subordinada puede ser una capa cliente desde la perspectiva de una tercera capa subordinada.
9. El software en las capas clientes y subordinadas puede físicamente compartir e intercambiar datos. Sin embargo, el software en cada capa interpreta los datos diferentemente.
10. El software que comparte datos con otro software no puede ser considerado en una capa diferente si interpreta los datos de manera idéntica al otro software.

[23] El concepto de capas presentado en este manual es diferente del concepto de "arquitectura por capas". Aunque existen elementos comunes a ambos conceptos, las capas COSMIC-FFP constituyen un instrumento de ayuda a la identificación de fronteras en la práctica. Si un paradigma arquitectural es utilizado en la organización, una equivalencia debe ser entonces establecida entre los objetos específicos de la arquitectura en este paradigma y el concepto de capas definido en este manual.

Una vez identificada, cada capa puede ser registrada en una línea individual dentro de la matriz del modelo genérico del software (Anexo A) con su identificación correspondiente. Ampliaciones sobre el concepto de capa figuran en el Anexo D.

REGLAS – Capas

a) Los servicios funcionales de los "paquetes software", tales como los sistemas de gestión de bases de datos, las interfases gráficas, los sistemas operativos o los dispositivos pilotos ("device drivers") son considerados generalmente como capas distintas.
b) Si un software es concebido utilizando un paradigma arquitectural conocido, ese paradigma debe ser utilizado para la identificación de las capas.
c) Un software de aplicación es considerado generalmente como la capa de más alto nivel.
d) En caso de duda, utilizar el concepto de acoplamiento para distinguir entre capas.

3.2 IDENTICACIÓN DE LAS FRONTERAS DEL SOFTWARE

Esta etapa consiste a la identificación de la frontera del software a ser medido. Una vez que la frontera ha sido indentificada, cada FUR describiendo procesos al interior de la frontera es incluído en el ámbito de la medición de la talla funcional.

DEFINICIÓN – Frontera

La frontera de una pieza de software es la frontera conceptual entre esta pieza y el ambiente en el cual opera, tal cual es percibido externamente desde la perspectiva de los usuarios. La frontera permite al medidor de distinguir, sin ambigüedad, lo que es parte del software, de lo que es parte de su ambiente operativo.

DEFINICIÓN – Usuarios

Seres humanos, dispositivos de ingeniería u otros software que interactúan con el software medido.

La identificación de la frontera es un proceso iterativo. La frontera exacta será afinada con el progreso del proceso de representación. Una vez identificada, la frontera candidata debe cumplir el principio siguiente:

PRINCIPIO – Frontera

a) Por definición, hay una frontera entre cada par de capas adyacentes y hay una frontera entre distintas piezas de software en la misma capa si los intercambios de datos entre ellas son del mismo nivel (comunicaciones "peer-to-peer").

Las siguientes reglas pueden ser útiles para confirmar el status de las candidatas a frontera:

REGLAS – Frontera

a) Comenzar identificando los eventos desencadenadores, luego identificar los procesos funcionales puestos en marcha por esos eventos. La frontera se sitúa entre esos eventos y los procesos funcionales.
b) Identificar los dispositivos de entrada y salida (E/S) utilizados por el software para interactuar con cada tipo de usuario. Relacionar esos dispositivos con las funciones provistas por el software. La frontera se sitúa entre esas funciones y los dispositivos de entrada y salida.
c) Para el software técnico o en tiempo real, utilizar el concepto de capas para ayudar a la identificación de las fronteras (sección 3.1).

A PROPÓSITO DE LA IDENTIFICACIÓN DE UNA FRONTERA Y DE LA UTILIZACIÓN DE CAPAS

El siguiente ejemplo ilustra una manera típica de identificación de una frontera de software con la ayuda del concepto de capa. La primera etapa es la identificación de los movimientos de datos a través la interfase de entrada y salida (E/S) con los usuarios. La frontera se encuentra entre los dispositivos E/S) y el software (Figura 3.2.1).

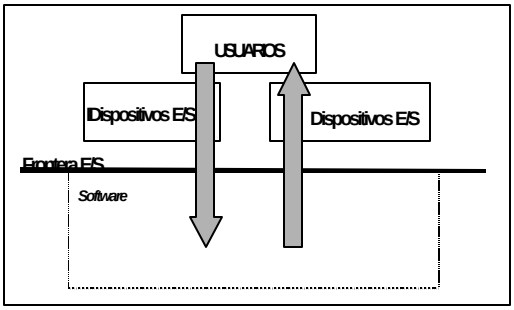


Figura 3.2.1 – Ilustración de la fronteras de entradas y salidas (E/S)

La siguiente etapa es la identificación de los movimientos de datos de y hacia los dispositivos de almacenaje, como están ilustrados en la figura 3.2.2.

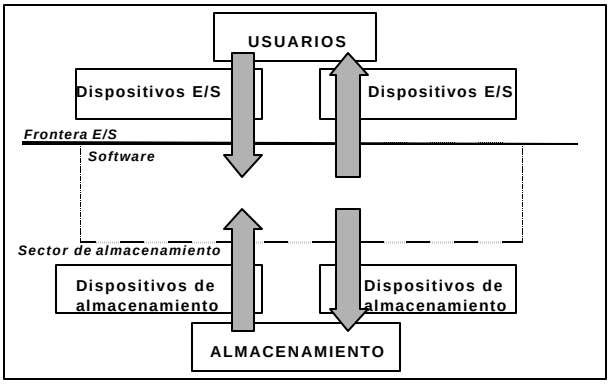


Figura 3.2.2 – Ilustración de la frontera de almacenamiento

Si una demanda de medición permite distinguir los procesos incluidos entre la frontera E/S y la frontera de almacenamiento, entonces el concepto de capa es utilizado para afinar la partición. Supongamos, por ejemplo, el objetivo de la medición es que los requerimientos funcionales del usuario asignados a un

dispositivo especial de almacenamiento se distinga de los otros FURs. El concepto de capa nos llevaría a la identificación de una frontera entre dos capas: la capa de aplicación y la capa piloto tal como ilustrado en la Figura 3.2.3. En este caso, la capa de aplicación es considerada como un usuario de la capa piloto. Hay que destacar que cuando los movimientos de datos entre dos capas son considerados, una "escritura" en la capa de aplicación, tal cual es vista por el usuario de la aplicación, es considerada como una "entrada" en la capa piloto y, consecuentemente, una "lectura" en la capa de aplicación, tal cual es vista por el usuario de la aplicación, es considerada como una "salida" en la capa piloto. Los movimientos de datos no son duplicados ya que la funcionalidad específica debe ser provista dentro de cada una de las dos capas para manipular dichos movimientos. La funcionalidad es así tomada en cuenta para cada capa.

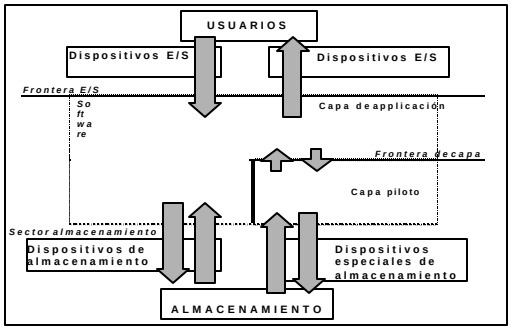


Figura 3.2.3 – Ilustración de frontera inter-capas

Más fronteras deberían ser identificadas de esta manera, sobre la base del objetivo de la medición y de la arquitectura específica del software. Cada capa resulta un usuario de las capas que directamente la proveen de los servicios funcionales requeridos.

3.3 IDENTIFICACIÓN DE PROCESOS FUNCIONALES

Esta etapa consiste en la identificación del conjunto de procesos funcionales que brindan las funcionalidades del software medido de acuerdo a los requerimientos. El ámbito de la medición de la talla funcional corresponde a todos los FUR que describen los procesos funcionales identificados.

DEFINICIÓN – Procesos functionales

Un proceso funcional es un conjunto único de movimientos de datos (entrada, salida, lectura, escritura) que implementan un conjunto coherente y lógicamente indivisible de requerimientos funcionales del usuario. Este proceso es desencadenado directamente, o indirectamente por intermedio de un "actor", por un evento (-tipo) y es completado cuando es ejecutado todo lo que se requiere hacer, en respuesta al evento (-tipo) que lo ha desencadenado.

DEFINICIÓN – Evento desencadenador

Un evento desencadenador ocurre al de la frontera del software e inicia uno o varios procesos. Los relojes y los eventos temporales pueden ser eventos desencadenadores. Cuando cada capa identificada es separada por una frontera, los eventos desencadenadores pueden ocurrir en una capa e iniciar procesos funcionales que pertenecen a otra capa.

Una vez identificados, los procesos funcionales candidatos deben cumplir los principios siguientes:

PRINCIPIOS – Procesos functionales

a) Un proceso funcional es una forma derivada de al menos un FUR.
b) Un proceso funcional es realizado cuando ocurre un evento desencadenador identificable.
c) Un proceso funcional contiene por lo menos dos movimientos de datos: una entrada y una salida o una escritura.
d) Un proceso funcional contiene no más de un estado de espera inducido (el cual puede ocurrir cuando es completado)
e) Un proceso funcional pertenece a una y sólo a una capa.

Las siguientes reglas son útiles para confirmar el status de los procesos funcionales candidatos:

REGLAS – Procesos functionales

a) Los sub-conjuntos de eventos desencadenadores no son considerados como eventos desencadenadores diferentes.

   Por ejemplo, si la ocurrencia de un evento específico desencadena la entrada de un grupo de datos incluyendo los atributos A, B y C, y que otra ocurrencia del mismo evento desencadena una entrada de un grupo de datos que contiene valores para los atributos A y B solamente, esta última ocurrencia no es considerada como un evento diferente. Esta ocurrencia es así considerada como la misma para la identificación de los procesos funcionales COSMIC-FFP. Como consecuencia, son identificados solamente una entrada y un proceso funcional que manipula los atributos de datos A, B y C.

   En el contexto del software en tiempo real, un proceso funcional es también un evento desencadenador. Él termina cuando un punto de tiempo asincrónico es alcanzado. Ese punto es alcanzado cuando, en una secuencia de movimientos de datos, un movimiento dado de datos no es sincronizado con el precedente. Un punto de tiempo sincrónico es equivalente a un estado de espera auto-inducido.

Una vez identificado, cada proceso funcional puede ser registrado sobre una línea en la capa apropiada de la matriz del modelo genérico de software (Anexo A), bajo la etiqueta correspondiente.

3.4 IDENTIFICACIÓN DE LOS GRUPOS DE DATOS

Esta etapa consiste en la identificación de los grupos de datos del software a ser medido.

DEFINICIÓN – Grupo de datos

Un grupo de datos es un conjunto distinto, no nulo, no ordenado y no redundante de atributos de datos en el cual cada dato incluido describe un aspecto complementario del mismo objeto de interés (ver definición). Un grupo de datos es caracterizado por su persistencia (ver definición).

DEFINICIÓN – Persistencia de grupos de datos[24]

La persistencia de un grupo de datos es una cualidad que describe la longitud del período de retención del grupo de datos en el contexto de los requerimientos funcionales del usuario. Tres tipos de persistencia son definidos:

. **Transitoria:** El grupo de datos no sobrevive a la transacción que los utiliza.
. **Corta**: El grupo de datos sobrevive a la transacción, pero no sobrevive a la operación del software.
. **Indefinida:** El grupo de datos sobrevive a la operación del software.

En la práctica, COSMIC FFP distingue sólo entre los grupos de datos transitorios y persistentes (i.e. persistencia corta o indefinida)

La persistencia de datos es una característica utilizada para ayudar a distinguir entre los cuatro tipos de sub-procesos, como será explicado en la sección 4 de este manual. Una vez identificado, cada grupo de datos, el candidato debe cumplir con los principios siguientes:

PRINCIPIOS – Grupo de datos

a) Un grupo de datos debe ser implantado en el sistema informático que soporta el software.
b) Cada grupo de datos identificado DEBE ser único y distinguible a través la sola colección de atributos de datos.
c) Los grupos de datos son directamente relacionados a los objetos en los FURs.

Una vez identificados, cada grupo de datos puede ser registrado en una columna individual de la matriz del modelo genérico de software (Anexo A), bajo la etiqueta correspondiente.

SOBRE LA IMPLANTACIÓN DE LOS GRUPOS DE DATOS

En la práctica, la implantación de los grupos de datos se realiza bajo tres formas, tal que ilustrado en la figura 3.4.1:

a) A través de una estructura física sobre un dispositivo permanente de almacenamiento (fichero, tabla, memoria ROM, etc.); esta forma de grupo de datos tiene una persistencia corta o indefinida.

b) A través una estructura física dentro de la memoria volátil de la computadora (estructura de datos asignada dinámicamente o a través un bloque de pre-asignado de espacio de memoria); esta forma tiene una persistencia corta o transitoria.

c) A través de la presentación conjunta de las funcionalidades que relacionan los atributos de datos a los dispositivos de E/S (pantalla, impresión, etc.), esta forma de grupo de datos tiene una persistencia variable, que depende de la utilización funcional de los datos y del medio usado por el dispositivo.

[24] Adaptado de la primera versión de ISO/IEC 14143/1: Software engineering – Software measurement – Functional size measurement method – Part 5: Definition of Functional Domains.

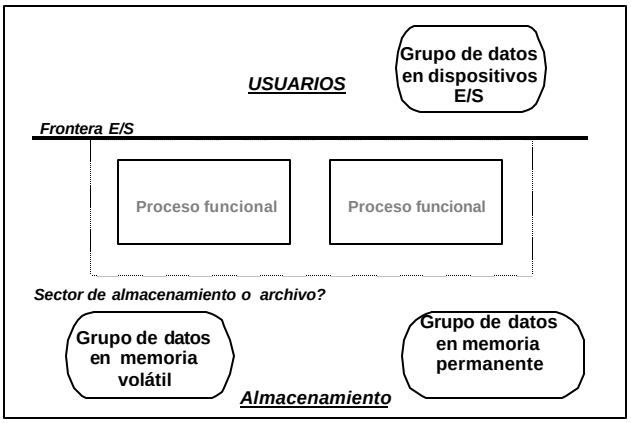


Figura 3.4.1 – Grupos de datos COSMIC-FFPy su relación con otros aspectos del modelo de software COSMIC-FFP

SOBRE LA IDENTIFICACIÓN DE UN GRUPO DE DATOS

La definición de los grupos de datos y sus principios son intencionalmente generales a fín que sean aplicables al más amplio rango posible de software, lo que tiene como inconveniente el hecho de hacer más difícil la medición de una pieza específica de software. Las regles siguientes, provenientes de la práctica de medición de la talla funcional, pueden ayudar a la aplicación de los principios en casos específicos:

REGLAS – Grupo de datos

a) APLICACIÓN A LOS SOFTWARES MIS (de gestión). Un grupo de datos es identificado para cada tipo de entidad observado en el modelo de datos normalizado[25] del software medido. Usualmente, los grupos de datos identificados tienen una persistencia indefinida y el software es requerido para almacenar los datos de las entidades-tipo concernidas.

Ejemplos: en el campo del software de información de gestión, un objeto puede ser "empleado" (físico) o "pedido" (conceptual) – el software es requerido para almacenar los datos de los empleados y de los pedidos.

Además, los grupos de datos que tienen una persistencia transitoria son formados cuando no existe demanda de datos sobre alguna cosa sobre la cual no hay datos almacenados con persistencia indefinida pero que pueden ser derivados de otros datos archivados con persistencia indefinida. En dichos casos, el objeto transitorio es objeto de un movimiento de entrada en una demanda ad-hoc (la selección de parámetros para derivar los datos requeridos) y del movimiento de datos de salida que contiene los atributos del objeto transitorio.

Ejemplo: se formula una demanda a la base de datos del personal para extraer la lista de los nombres de todos los empleados de más de 35 años. Este grupo es un objeto transitorio. La entrada es un grupo de datos que contiene los parámetros. La salida es un grupo de datos que contiene la lista de nombres.

[25] Segunda o tercera forma normal.

REGLAS – Grupo de datos

b) APLICACIÓN A LOS SOFTWARES EN TIEMPO REAL. La práctica de medición ha establecido que los grupos de datos para este tipo de software tienen muchas veces las formas siguientes:

- Los movimientos de datos que son entradas de dispositivos físicos, típicamente contienen datos sobre el estado de un objeto (-tipo) simple o pueden ser el objeto mismo. Ejemplo: en un intercambio de mensajes, el objeto puede ser un mensaje que el software requiere para operar, e.g. enviar o almacenar y enviar, directamente.

- Estructura de datos comunes, representando objetos que son mencionados en los requerimientos funcionales del usuario, los que son guardados en la memoria volátil y accesible a la mayor parte de procesos funcionales del software medido.

- Estructura de datos referenciales, representados bajo la forma de gráficos o tablas en los requerimientos funcionales del usuario, guardados en memoria permanente y accesibles a la mayor parte de los procesos funcionales del software medido.

- Almacenamientos simples, representando objetos que figuran en los requerimientos funcionales del usuario que son conservados en dispositivos de memoria permanente.

EEMPLOS DE IDENTIFICACIÓN DE GRUPOS DE DATOS

Esta sección presenta dos ejemplos MIS con el fin de ilustrar cómo los grupos de datos son identificados a partir de los requerimientos funcionales del usuario

EJEMPLO 1

Consideremos los requerimientos funcionales del usuario siguientes:

"Los datos permanentes sobre los empleados y los departamentos de la organización deben ser mantenidos. A partir de estos datos, el software debe ser capaz de proveer la historia del empleo de un empleado, es decir, la lista de departamentos en los cuales este empleado ha trabajado en el pasado" .

Analicemos los grupos de datos mencionados en este requerimiento. Los dos grupos más obvios son "Empleado" y "Departamento" , tal cual es mostrado en la Figura 3.4.2. Existe también una relación implícita entre estos dos grupos de datos: un empleado puede estar ligado a varios departamentos y un departamento puede contener varios empleados. Esta relación es igualmente ilustrada en la Figura 3.4.2.

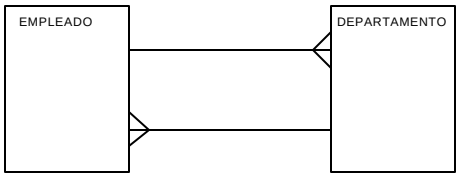


Figura 3.4.2 – Ejemplos de grupos de datos con una relación de "varios a varios"

En la práctica, la relación tiene sus propios atributos (la fecha de llegada de un empleado a un departamento y la de su salida). Esta relación constituye un grupo de datos en sí, y según las prácticas de ingeniería de software, ella es materializada en una estructura de datos independiente (ver Figura 3.4.3).

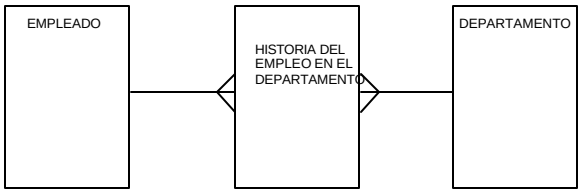


Figura 3.4.3 – Ejemplos de grupos de datos tal cual identificados por COSMIC-FFP

Con el objetivo de medir la talla funcional de este FUR, las reglas de medición COSMIC-FFP identificarían tres grupos de datos distintos en este ejemplo, los que serían caracterizados por su persistencia indefinida.

EJEMPLO 2

Consideremos el requerimiento funcional del usuario siguiente:

"El software debe considerar una interrogación ad-hoc a la base de datos del personal a fin de extraer la lista de los nombres de todos los empleados mayores de 35 años".

Analicemos los grupos de datos mencionados en este requerimiento. El grupo de datos mas obvio es "Empleados", que es caracterizado (implícitamente) por su persistencia indefinida. La selección de parámetros, provista por una entrada, son los atributos de un segundo grupo de datos, caracterizado por una no persistencia. Un tercer grupo de datos, también caracterizado por su no persistencia, es formado por los resultados de la interrogación y mostrado en pantalla enseguida de la interrogación, mientras que él caracteriza otro objeto de interés que podría ser etiquetado como "resultados de la interrogación". Esos tres grupos de datos, con sus movimientos respectivos, son ilustrados en la Figura 3.4.4.

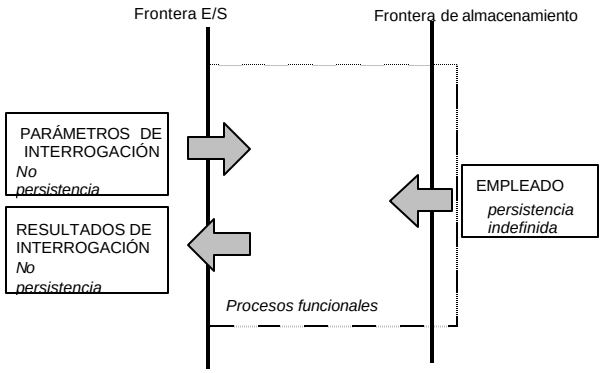


Figura 3.4.4 – Ejemplos de grupos de datos identificados tal cual por COSMIC-FFP

3.5 IDENTICACIÓN DE LOS ATRIBUTOS DE DATOS

Esta etapa consiste a identificar de conjunto de atributos referidos al software a ser medido. En esta versión del método de medición, esta identificación no es obligatoria. Los atributos pueden ser identificados si una sub-unidad de medida es requerida, tal cual es presentado en la sección 2.4 (característica 4).

DEFINICIÓN – Atributos de datos
Un atributo de datos es la parcela más pequeña de información, dentro de un grupo de datos identificado, que tiene un significado desde la perspectiva del FUR del software.

DEFINICIÓN – Tipos de atributos de datos

Entre todos los atributos de datos referenciados por el software a ser medido, se puede distinguir tres tipos, de los cuales, dos son válidos para la medición de la talla funcional del software.

TIPOS DE DATOS VÁLIDOS

- **Los datos que describen o representan el mundo del usuario**. Este tipo es independiente de su procesamiento. Un nombre de empleado y su salario tienen significado sin considerar el cómo ellos serán procesados por un conjunto de procesos funcionales, como por ejemplo, la impresión de cheques de pago.
- **Datos contextuales**. Este tipo especifica qué, cuándo y cómo otros datos son procesados por el software medido. Su significado funcional es completamente comprendido cuando se considera el contexto en el cual ellos son referenciados. Supongamos el requerimiento funcional del usuario siguiente: "*imprimir el informe XYZ a las 10:00*". La hora especificada (10.00) no tiene significación funcional sin considerar el contexto del "informe XYZ", que determina cuando ese informe debe ser impreso.

TIPOS DE DATOS NO VALIDOS

- **Implementación los datos técnicos**. Este tipo de dato es creado solamente porque es requerido por la técnica escogida para implantación del software.

Una vez identificados, los procesos funcionales candidatos deben cumplir los principios siguientes:

PRINCIPIOS – Atributos de datos

a) Un atributo de datos debe representar un tipo válido de datos.
b) Un atributo de datos debe representar la más pequeña pieza de datos referida por el software medido, desde la perspectiva de los requerimientos funcionales del usuario.

EJEMPLOS – Atributos de datos

a) EN EL CONTEXTO DE LAS APLICACIONES SOFTWARE MIS
Los tipos de elementos de datos registrados en el diccionario de datos,
Los atributos de datos que aparecen en un modelo de datos conceptual o lógico.

b) EN EL CONTEXTO DE LAS APLICACIONES SOFTWARE EN TIEMPO REAL
Una señal de voltaje recibida por un censor.

SOBRE LA ASOCIACIÓN DE LOS ATRIBUTOS Y GRUPOS DE DATOS

Dentro del contexto de los requerimientos funcionales del usuario del software, hay una distinción entre un objeto de interés y los datos necesarios a describirlo; el objeto es representado por el grupo de datos, mientras que los datos son representados por los atributos. Así, en teoría, un grupo de datos puede contener sólo un atributo si esto es suficiente, desde la perspectiva de los requerimientos funcionales, para describir el objeto de interés. En la práctica, ese caso ha sido observado en los software en tiempo real, pero no ha sido nunca observado todavía en los software de tipo MIS.

4

FASE DE MEDICIÓN – REGLAS Y MÉTODO

El método de medición COSMIC-FFP considera la medición de la talla funcional del software a través dos fases distintas: la representación del software a ser medido en el modelo genérico de software COSMIC-FFP y la medición de los aspectos específicos de este modelo.

La derivación de la talla funcional del software a ser medido es independiente del esfuerzo requerido para desarrollarlo o mantenerlo, del método usado para desarrollarlo o mantenerlo y de cualquier componente físico o tecnológico del software.

Esta sección presenta las reglas y el método para la fase de medición. El método general de medición de una pieza de software a partir del modelo genérico de software COSMIC-FFPes resumido en la Figura 4.1.

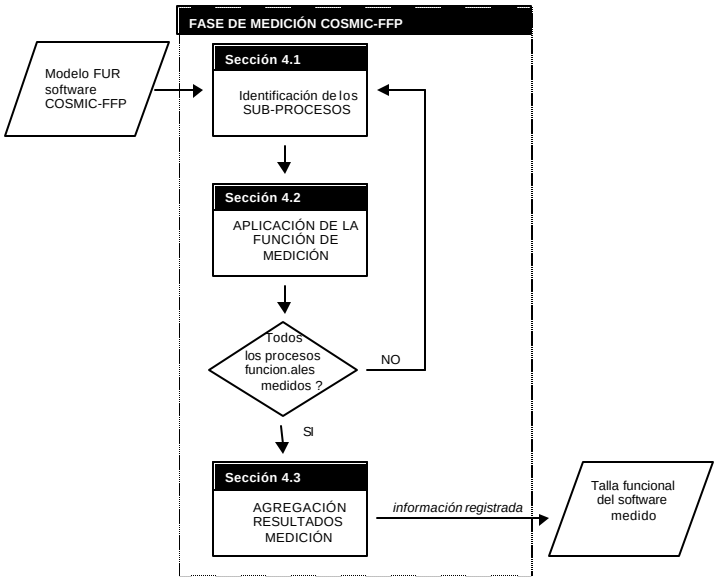


Figura (Tabla) 4.1 – Procedimiento general para la fase de medición COSMIC-FFP

Cada etapa en este método es el objeto de una sub-sección específica, en la cual las definiciones y los principios de aplicación son presentados, con algunas reglas y ejemplos. Un resumen de los principios y reglas figura en los Anexos B y C respectivamente.

4.1 IDENTIFICACIÓN DE LOS SUB-PROCESOS

Esta etapa consiste en la identificación de los sub-procesos de cada proceso funcional. Ella es aplicada a cada uno de los procesos a ser identificados en el software a ser medido. Una vez que los sub-procesos sean identificados, sus tipos serán identificados de acuerdo a los procedimientos definidos en las páginas siguientes.

DEFINICIÓN – Sub-proceso COSMIC-FFP

Un sub-proceso COSMIC-FFP es un movimiento de datos que ocurre durante la ejecución de un proceso funcional. Hay cuatro sub-tipos de sub-procesos COSMIC FFP : entrada, salida, lectura y escritura, cada uno de los cuales incluye sub-procesos específicos de manipulación de datos. Un sub-proceso COSMIC-FFP es equivalente a un Tipo de Componente Funcional de Base, tal cual es considerado por ISO. Un sub-proceso COSMIC-FFP expresa sólamente los requerimientos funcionales del utilizador y excluye los requerimientos técnicos y de calidad.

DEFINICIÓN – Entrada (E)

Una ENTRADA (E) es un movimiento de datos que se encuentra en un grupo de datos desde el lado del usuario de la frontera hacia el interior de la frontera del software. Una ENTRADA no actualiza los datos que ella desplaza. Funcionalmente, un sub-proceso de ENTRADA trae los datos, que se encuentran del lado del utilizador, al proceso funcional al cual ella pertenece. Hay que notar que en el COSMIC FFP, una ENTRADA incluye algunos sub-procesos de manipulación de datos.

DEFINICIÓN – Salida (X)

Una SALIDA (X) es un movimiento de atributos de datos que se encuentran en un grupo de datos dentro de la frontera del software hacia el lado del usuario de la frontera del software. Una SALIDA no lee los datos, sólo los desplaza. Funcionalmente, una SALIDA envía datos desde el proceso funcional al cual pertenece (implícitamente al interior de la frontera del software) hacia el lado del usuario de la frontera. Hay que notar que en el COSMIC FFP, una SALIDA incluye algunos sub-procesos de manipulación de datos.

DEFINICIÓN – Lectura (R)

Una LECTURA (R) se refiere a los atributos de datos encontrados en un grupo de datos. Funcionalmente, un sub-proceso de LECTURA trae los datos del archivo (almacenamiento) al proceso funcional al cual pertenece. Hay que notar que en COSMIC FFP, una LECTURA incluye algunos sub-procesos de manipulación asociados de datos.

DEFINICIÓN – Escritura (W)

Una ESCRITURA (W) se refiere a los atributos de datos encontrados en un grupo de datos. Funcionalmente, un sub-proceso de ESCRITURA envía los datos desde el proceso funcional al que pertenece hacia el archivo (almacenamiento). Hay que notar que en COSMIC FFP, una ESCRITURA incluye algunos sub-procesos de manipulación asociados de datos.

La Figura 4.1.1, ilustra las diferentes relaciones entre los cuatro tipos de sub-procesos, el proceso funcional al cual ellos pertenecen y la frontera del software medido.

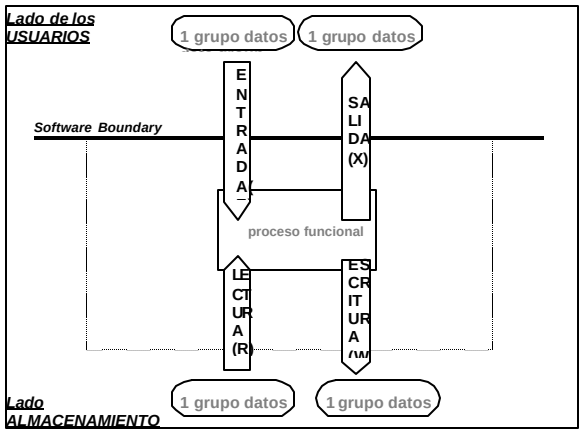


Figure 4.1.1 – Los cuatro tipos de sub-procesos y sus relaciones con el proceso funcional y los grupos de datos

ÁMBITO DE LAS MANIPULACIONES DE DATOS ASOCIADA A LOS SUB-PROCESOS

Por definición, los sub-procesos son movimientos funcionales de datos elementales o manipulaciones de datos que ocurren dentro de la frontera del software medido (según la convención COSMIC FFP, que no reconoce la existencia separada de sub-procesos de manipulación de datos). Dependiendo de su tipo, los sub-procesos representan, por convención, las siguientes funcionalidades detalladas:

SUB-PROCESO DE ENTRADA

Una entrada incluye todas las manipulaciones de formateo y presentación requeridas por los usuarios así como todas las validaciones asociadas a la entrada de datos, siempre y cuando dichas manipulaciones no impliquen otro tipo de sub-procesos. Por ejemplo, una entrada incluye todas esas manipulaciones, EXCEPTO las lecturas requeridas para validar algunos códigos o algunas descripciones asociadas.

SUB-PROCESO DE SALIDA

Una salida incluye todas las manipulaciones de formateo y presentación requeridas por los usuarios, incluido el procesamiento requerido para enviar los resultados a esos usuarios, siempre y cuando dichas manipulaciones no impliquen otro tipo de sub-procesos. Por ejemplo, una salida incluye todas los tratamientos de formateo y preparación para la impresión de atributos de datos, EXCEPTO las lecturas o entradas requeridas para proveer valores (o descripciones) de alguno de los atributos de datos.

SUB-PROCESO DE LECTURA

Una lectura incluye todos los tratamientos y/o cálculos asociados a la lectura de datos, siempre y cuando dichas manipulaciones no impliquen otro tipo de sub-procesos. Por ejemplo, una lectura incluye todos los cálculos matemáticos y tratamientos lógicos requeridos para adquirir atributos de un grupo de datos, EXCEPTO las salidas requeridas para la impresión o el afichaje de esos atributos de datos.

SUB-PROCESO DE ESCRITURA

Una escritura incluye todos los tratamientos y/o cálculos necesarios al almacenamiento de atributos de datos, siempre y cuando dichas manipulaciones no impliquen otro tipo de sub-procesos. Por ejemplo, una escritura incluye todos los cálculos y tratamientos lógicos necesarios para actualizar un grupo de datos, EXCEPTO las lecturas o entradas requeridas para proveer valores de otros atributos de datos incluidos en la actualización del grupo de datos par la escritura.

4.1.1 Identificación de una Entrada (E)

Una vez identificado, un subproceso candidato a ENTRADA debe satisfacer los principios siguientes:

PRINCIPIOS – Entrada (E)

a) El sub-proceso recibe atributos de datos desde fuera de la frontera del software, desde el lado del usuario.
b) El sub-proceso recibe datos de sólo un grupo de datos. Si el input proviene de más de un grupo de datos, entonces identificar una ENTRADA por cada grupo.
c) El sub-proceso no extrae, lee ni escribe datos.
d) Dentro del proceso funcional donde es identificado, el sub-proceso es único, es decir, el procesamiento y los atributos de datos identificados son diferentes de los de otras entradas incluidas en el mismo proceso funcional.

Las reglas siguientes son útiles para confirmar el status de un sub-proceso candidato a entrada:

REGLAS – Entrada (E)

a) Los eventos desencadenados por reloj son considerados como externos. Por ejemplo, un evento que ocurre cada 3 segundos es asociado con una ENTRADA que mueve un atributo de datos. Sin embargo, el proceso funcional que genera el evento periódicamente es ignorado ya que él ocurre, por definición, fuera de la frontera del software.
b) A menos que un proceso funcional específico sea necesario, la obtención del tiempo en el reloj no es considerada como una ENTRADA. Por ejemplo, cuando un proceso funcional escribe un sello de tiempo, ninguna ENTRADA es identificada por la obtención del valor del tiempo en el sistema del reloj.

Una vez identificado, cada sub-proceso de entrada puede ser registrado marcando la celda correspondiente en la matriz del modelo genérico de software (Anexo A) con una "E".

4.1.2 Identificación de una Salida (X)

Una vez identificado, el sub-proceso candidato a SALIDA debe satisfacer los principios siguientes:

PRINCIPIOS – Salida (X)

a) El sub-proceso envía atributos de datos fuera de la frontera del software, del lado del usuario.
b) El sub-proceso envía datos que pertenecen sólo a un grupo de datos. Si los datos pertenecientes a más de un grupo son enviados al exterior de la frontera, identificar una SALIDA por cada grupo de datos referenciado.
c) El sub-proceso no recibe, lee ni escribe datos.
d) Dentro del proceso funcional en el cual es identificado, el sub-proceso es único, es decir, el tratamiento y los datos identificados son diferentes de los de otras SALIDAS incluidas en el mismo proceso funcional.

La regla siguiente puede ser útil para confirmar el status de un sub-proceso candidato a salida.

REGLAS – Salida (X)

a) Todos los mensajes generados que no contengan datos del usuario (i.e. mensajes de error o confirmación) son identificados como una SALIDA simple del proceso funcional en el cual la salida es identificada.

   Por ejemplo, supongamos los procesos funcionales A y B identificados en la misma capa. "A" puede potencialmente emitir 2 (tipos) de mensajes de confirmación y 5 mensajes de error a los usuarios, y "B" puede potencialmente emitir 8 mensajes de error a los usuarios. En este ejemplo, una SALIDA sería identificada en el proceso funcional "A" (manipulando 5+2=7 mensajes) y una SALIDA separada sería identificada en el proceso funcional "B" (manipulando 8 mensajes).

Una vez identificada, cada sub-proceso de SALIDA puede ser registrado marcando la celda correspondiente en la matriz del modelo genérico de software (Anexo A) con una "X".

4.1.3 Identificación de una Lectura (R)

Una vez identificado, el sub-proceso candidato a LECTURA debe satisfacer los principios siguientes:

PRINCIPIOS – Lectura (R)

a) El sub-proceso recupera los atributos de datos en el grupo de almacenamiento de datos.
b) El sub-proceso refiere atributos que pertenecen a UN sólo grupo de datos. Si más de un grupo de datos es referenciado, identificar una LECTURA por cada grupo de datos.
c) El sub-proces no recibe, extrae ni escribe datos.
d) En el proceso funcional dentro del cual es identificado, el sub-proceso es único, es decir, el tratamiento y los atributos de datos identificados son diferentes de los de otras LECTURAS incluidas en el mismo proceso funcional.

Una vez identificada, cada sub-proceso de LECTURA puede ser registrado marcando la celda correspondiente en la matriz del modelo genérico de software (Anexo A) con una "R".

4.1.4 Identificación de una Escritura (W)

Una vez identificado, el sub-proceso candidato a ESCRITURA debe satisfacer los principios siguientes:

PRINCIPIOS – Escritura (W)

a) El sub-proceso graba los atributos de datos de un grupo de datos en el lado de almacenamiento del software.
b) El sub-proceso modifica el valor de los atributos de datos que pertenecen a sólo UN grupo de datos. Si más de un grupo de datos es referenciado, identificar una ESCRITURA por cada grupo de datos.
c) El sub-proceso no recibe, extrae ni lee datos.
d) En el proceso funcional dentro del cual es identificado, el sub-proceso es único, es decir, el tratamiento y los atributos de datos identificados son diferentes de los de otras ESCRITURAS incluidas en el mismo proceso funcional.

Una vez identificada, cada sub-proceso de ESCRITURA puede ser registrado marcando la celda correspondiente en la matriz del modelo genérico de software (Anexo A) con una "W".

4.1.5 Extensiones locales

El método de medición COSMIC-FFP no prevé una manera standard para medir la talla funcional de ciertos tipos de requerimientos funcionales del usuario, como los requerimientos que contienen algoritmos matemáticos complejos o secuencias de reglas complejas como las que se encuentran normalmente en los sistemas expertos. Sin embargo, unidades de medida locales, que corresponden a las organizaciones en las que se aplica el método COSMIC-FFP, podrían ser utilizados a fin de implementar la medición de las funcionalidades del software.

Por esta razón, el método de medición COSMIC-FFP considera la utilización de extensiones locales del método. En algunos procesos, en los cuales existen sub-procesos funcionales de manipulación anormalmente complejos, el medidor es libre de asignar sus propias unidades de talla funcional localmente determinadas para dichas funcionalidades excepcionales. Un ejemplo de standard de extensión local podría ser:

> *En nuestra organización, nosotros asignamos una unidad de talla funcional para los algoritmos matemáticos como (lista de significaciones locales y ejemplos bien comprendidos…). Nosotros asignamos dos unidades para (otra lista de ejemplos), etc.*

Cuando dichas extensiones locales sean utilizadas, los resultados de la medición deben ser informados de acuerdo a la convención especial presentada en la sección 5 de este manual.

4.2 APPLICACIÓN DE LA FUNCIÓN DE MEDICIÓN

Esta etapa consiste en la aplicación de la función de medición COSMIC-FFP a cada uno de los sub-procesos identificados en cada proceso funcional.

DEFINICIÓN – Función de medición COSMIC-FFP

La función de medición COSMIC-FFP es una función matemática que asigna un valor a una variable sobre la base del standard de medición COSMIC-FFP. La variable de la función de medición COSMIC-FFP es el sub-proceso.

DEFINICIÓN – Unidad de medida COSMIC-FFP

El standard de medición COSMIC-FFP, 1 Cfsu (Cosmic Functional Size Unit), es definido como un movimiento de datos elemental.

De acuerdo a esta función de medición, cada instancia de un sub-proceso (entrada, salida, lectura o escritura) identificada en la etapa 1 (Figura 4.1) tiene una talla numérica de 1 Cfsu.

4.3 AGREGATCIÓN DE LOS RESULTADOS DE LA FUNCIÓN DE MEDICIÓN

Esta etapa consiste en la agregación de los resultados de la función de medición, tal cual ella es aplicada a los sub-procesos identificados, en un sólo valor de talla funcional. Esta etapa es realizada de acuerdo a los principios y reglas siguientes:

PRINCIPIO – Agregación de los resultados de la medición

a) Para cada capa identificada, las tallas funcionales de los sub-procesos individuales son sumadas aritméticamente para obtener en un sólo valor de talla funcional.

$$Talla_{Cfsu}\ (capa_i) = \sum talla(entradas_i) + \sum talla(salidas_i) + \sum talla(lecturas_i) + \sum talla(escrituras_i)$$

b) Para cada capa identificada, la talla funcional de los cambios en los requerimientos funcionales del usuario es la suma de las tallas de los sub-procesos correspondientes modificados de acuerdo a la fórmula siguiente:

$$Talla_{Cfsu}\ (Cambio(capa_i)) = \sum talla(sub\text{-}procesos_i\ adicionados) + \sum talla(sub\text{-}procesos_i\ modificados) + \sum talla\ (sub\text{-}procesos_i\ eliminados\ )$$

REGLA – Agregación de los resultados de la medición

a) Cuando se utiliza la matriz de medición del Anexo A, la agregación debe ser realizada adicionando el número de sub-procesos de cada fila de una capa dada.

Hay que destacar que, para cada cosa identificada, la función de agregación es totalmente adaptable. Así, un sub-total puede ser generado para los procesos funcionales individuales o para toda la capa, dependiendo del objetivo y de la envergadura del ejercicio de medición.Además, la agregación de los resultados de la medición por tipo de sub-proceso, puede ser útil para analizar la contribución de cada tipo a la talla total de una capa, y puede ayudar también a caracterizar la naturaleza funcional de la capa medida.

SOBRE LA AGREGACIÓN DE LA TALLA FUNCIONAL

En un contexto en el cual la talla funcional es utilizada como variable en un modelo de estimación del esfuerzo, por ejemplo, y el software a ser medido tiene más de una capa, la agregación será realizada típicamente a nivel de capa, ya que las capas no son implementadas usualmente utilizando la misma tecnología. Por ejemplo, supongamos que el software en el cual la capa de aplicación es implementada utilizando un lenguaje de tercera generación y un conjunto de bibliotecas existentes, mientras que la capa piloto es implementada utilizando el lenguaje Assembler. El esfuerzo asociado a la construcción de cada capa será probablemente diferente, y consecuentemente, una estimación del esfuerzo será preparada de manera separada para cada capa sobre la base de su talla respectiva.

5

PRESENTACIÓN DE LA MEDICIÓN COSMIC-FFP

Los resultados de la medición COSMIC-FFP deben ser presentados y archivados de acuerdo a las convenciones siguientes.

5.1 ETIQUETAJE

La presentación de la talla funcional COSMIC-FFP total debe ser etiquetada de acuerdo a la convención siguiente, basada sobre la norma ISO 14143-1.

CONVENCIÓN[26] – Etiquetaje de la medición COSMIC-FFP

El resultado de la medición COSMIC-FFP debe ser denotado como "***x*** Cfsu (v. ***y***) ",donde :

- "**x**" representa el valor numérico obtenido a partir de la suma de todos los resultados individuales de la medición en una capa.
- "**y**" representa la identificación de la versión del método COSMIC-FFP utilizado en la obtención del valor numérico "x" de la talla funcional.

Cuando alguna extensión local ha sido utilizada, de acuerdo a la definición de la sección 4.1.5, los resultados de la medición deben ser presentados según la convención que figura a continuación.

CONVENCIÓN[27] – Etiquetaje de las extensiones locales COSMIC-FFP

El resultado de la medición COSMIC-FFP que utiliza extensiones locales es denotado así:

" ***x*** Cfsu (v. ***y***) + z Unidad de talla funcional local", donde:

- "**x**" representa el valor numérico obtenido la suma de todos los resultados individuales de medición, en una capa, de acuerdo con la versión v.y del método COSMIC-FFP,
- "**y**" representa la identificación de la versión del método COSMIC-FFP utilizado en la obtención del valor numérico "x" de la talla funcional.
- "**z**" representa el valor numérico obtenido por la suma de todos los resultados individuales de medición, que han utilizado las extensiones locales del método COSMIC FFP en una capa dada.

[26] De: Morris P., Desharnais J.-M., "*Measuring ALL the software, not just what the Business uses*", Proceedings of the IFPUG Conference, Orlando, 1998.

[27] De: Morris P., Desharnais J.-M., "*Measuring ALL the software, not just what the Business uses*", Proceedings of the IFPUG Conference, Orlando, 1998.

5.2 ALMACENAMIENTO DE LOS RESULTADOS DE LA MEDICIÓN COSMIC-FFP

El almacenamiento de los resultados de la medición COSMIC-FFP, debe considerar el registro de las informaciones siguientes.

CONVENCIÓN – Detalle de la medición COSMIC-FFP

- Un registro distinto por cada capa debe ser conservado. Ese registro contiene al menos la siguiente información:
  - Identificación del software medido (nombre, No. versión o No. configuración)
  - Una descripción de la capa,
  - El número total de procesos funcionales identificados,
  - El número total de grupos de datos identificados,
  - La talla funcional total de los sub-procesos de ENTRADA,
  - La talla funcional total de los sub-procesos de SALIDA,
  - La talla funcional total de los sub-procesos de LECTURA,
  - La talla funcional total de los sub-procesos de ESCRITURA.

6

CONVERTIBILIDAD COSMIC-FFP

Esta sección será completada en una próxima sub-versión del método de medición COSMIC-FFP.

6.1 Full Function Points, versión 1.0

Esta sección será completada en una próxima sub-versión del método de medición COSMIC-FFP.

6.2 Puntos de función IFPUG, versión 4.1

Esta sección será completada en una próxima sub-versión del método de medición COSMIC-FFP.

6.3 MarkII, versión 3.1.1

Esta sección será completada en una próxima sub-versión del método de medición COSMIC-FFP.

7

MEDICIÓN COSMIC-FFP PREVIA AL DESARROLLO

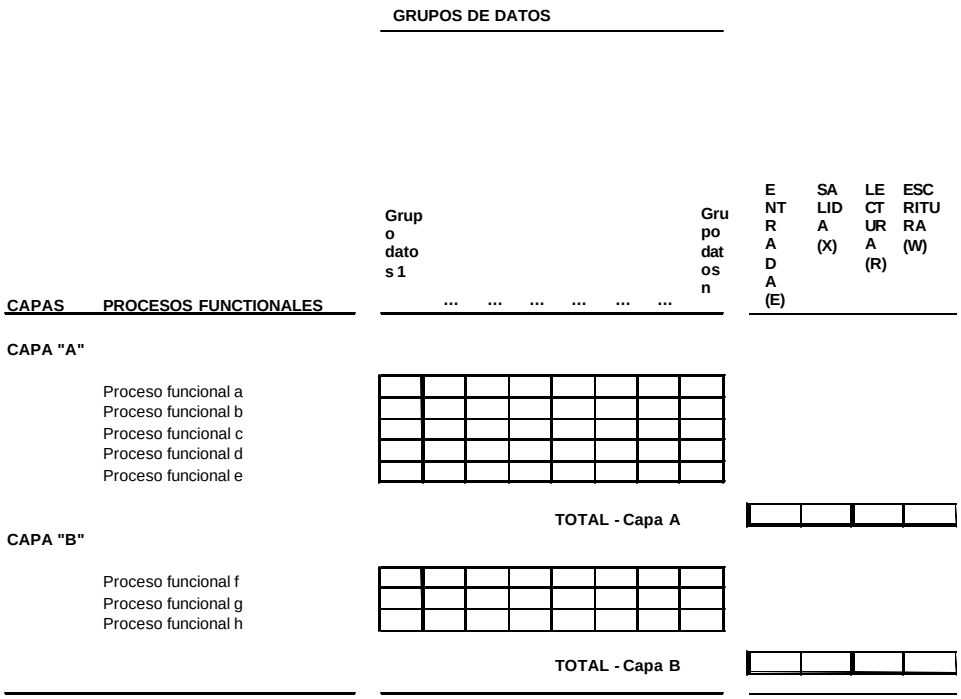
Esta sección será completada en una próxima sub-versión del método de medición COSMIC-FFP, sobre la base del trabajo del Dr. Roberto Meli.

ANEXOS

Anexo A

MODELO DE SOFTWARE COSMIC-FFP

La matriz mostrada debajo debe ser utilizada como un depósito que conserva cada componente identificado del modelo del software a ser medido. Ella ha sido diseñada para facilitar el proceso de medición.



FASE DE REPRESENTACIÓN

- Cada grupo de datos identificado es registrado en una columna,
- Cada proceso funcional es registrado en una línea específica y agrupados por capas identificadas.

FASE DE MEDICIÓN

- Para cada proceso funcional identificado, los sub-procesos identificados son anotados en la celda correspondiente utilizando la convención siguiente: “E” por una entrada, “X” por una salida, “R” por una lectura y “W” por una escritura;
- Para cada proceso funcional identificado, los sub-procesos son sumados por tipo, y cada sub-total es registrado en la columna apropiada, al lado derecho de la matriz;
- El resumen de la medición puede entonces ser calculado y registrado en las celdas de cada capa, en la línea de los totales.

Anexo B

PRINCIPIOS DE IDENTIFICACIÓN COSMIC-FFP

A fin de permitir referencias precisas, la siguiente tabla muestra los principios del método de COSMIC-FFP.

ID	NOMBRE	DESCRIPCIÓN
P-01	Capa	a) El software en todas las capas provée funcionalidades a los usuarios (Un usuario puede ser un ser humano, un dispositivo físico u otro software). b) El software en una capa subordinada provee servicios funcionales al software en las capas clientes. c) El software en las capas subordinadas puede ejecutarse sin la asistencia del software en una capa cliente. d) El software en una capa cliente puede no ejecutarse apropiadamente si el software en la capa subordinada que depende no es ejecutado apropiadamente. e) El software en la capa cliente no usa necesariamente todas las funcionalidades provistas por el software en una capa subordinada. f) El software en una capa subordinada puede ser una capa cliente desde la perspectiva de una tercera capa subordinada. g) El software en las capas clientes y subordinadas puede físicamente compartir e intercambiar datos. Sin embargo, el software en cada capa interpreta los datos diferentemente. h) El software que comparte datos con otro software no puede ser considerado en una capa diferente si interpreta los datos de manera idéntica al otro software.
P-02	Frontera	a) Por definición, hay una frontera entre cada par identificado de capas adyacentes y puede haber una frontera entre distintas piezas de software en la misma capa, si ellas intercambian datos mediante comunicaciones del mismo nivel ("peer-to-peer").
P-03	Proceso funcional	a) Un proceso funcional es derivado de al menos un requerimiento funcional del usuario. b) Un proceso funcional es realizado cuando un evento desencadenador identificable ocurre. c) Un proceso funcional contiene al menos dos movimientos de datos, una entrada y una salida o escritura. d) Un proceso funcional contiene no más que un estado de espera auto-inducido (el que puede ocurrir cuando es completado), e) Un proceso funcional pertenece a una y sólo a una capa.
P-04	Grupo de datos	a) Un grupo de datos debe ser implantado en el sistema computarizado que soporta el software. b) Cada grupo de datos identificado DEBE ser único y distinguible a través la colección única de atributos de datos. c) Los grupos de datos son directamente relacionados a los objetos

ID	NOMBRE	DESCRIPCIÓN
		descritos en los requerimientos funcionales del usuario del software.
P-05	Atributos de datos	a) Un atributo (de datos) debe representar un tipo válido de datos. b) Un atributo (de datos) debe representar la más pequeña pieza de datos referenciada por el software medido desde la perspectiva de los requerimientos funcionales del usuario.
P-06	ENTRADA	a) El sub-proceso recibe datos del exterior de la frontera del software, desde el lado del usuario. b) El sub-proceso recibe datos pertenecientes a un solo grupo de datos. Si el input de más de un grupo de datos es recibido, identificar una ENTRADA por cada grupo de datos. c) El sub-proceso no extrae, lee ni escribe datos. d) En el proceso funcional dentro del cual es identificado, el sub-proceso es único, es decir, los tratamientos y los atributos de datos identificados son diferentes de las otras ENTRADAS identificadas en el mismo proceso funcional.
P-07	EXIT	a) El sub-proceso envía datos al exterior de la frontera del software, del lado del usuario. b) El sub-proceso envía datos pertenecientes a un solo grupo de datos. Si datos pertenecientes a más de un grupo de datos son enviados fuera de la frontera, identificar una SALIDA por cada grupo de datos referenciado. c) El sub-proceso no recibe, lee ni escribe datos. d) En el proceso funcional dentro del cual es identificado, el sub-proceso es único, es decir, los tratamientos y los atributos de datos identificados son diferentes de las otras SALIDAS identificadas en el mismo proceso funcional.
P-08	LECTURA	a) El sub-proceso recupera atributos de datos desde un grupo de datos almacenados. b) El sub-proceso refiere datos pertenecientes a UN solo grupo de datos. Si más de un grupo de datos es referenciado, identificar una LECTURA por cada grupo de datos. c) El sub-proceso no recibe, extrae ni escribe datos. d) En el proceso funcional dentro del cual es identificado, el sub-proceso es único, es decir, los tratamientos y los atributos de datos identificados son diferentes de las otras LECTURAS identificadas en el mismo proceso funcional.
P-09	ESCRITURA	a) El sub-proceso almacena datos en un grupo de datos del lado de almacenamiento del software. b) El sub-proceso modifica datos pertenecientes a UN solo grupo de datos. Si más de un grupo de datos es referenciado, identificar una ESCRITURA por cada grupo de datos. c) El sub-proceso no recibe, extrae ni lee datos. d) En el proceso funcional dentro del cual es identificado, el sub-proceso es único, es decir, los tratamientos y los atributos de datos identificados son diferentes de las otras ESCRITURAS identificadas en el mismo proceso funcional.

ID	NOMBRE	DESCRIPCIÓN
P-10	Agregación de los resultados de la medición	a) Para cada capa identificada, las tallas funcionales de los sub-procesos individuales son sumadas aritméticamente para obtener en un sólo valor de talla funcional. $Talla_{Cfsu}\ (capa_i) = \sum talla(entradas_i) + \sum talla(salidas_i) + \sum talla(lecturas_i) + \sum talla(escrituras_i)$ b) Para cada capa identificada, la talla funcional de los cambios en los requerimientos funcionales del usuario, es la suma de las tallas de los sub-procesos correspondientes modificados, de acuerdo a la fórmula siguiente: $Talla_{Cfsu}\ (Cambio(capa_i)) = \sum talla(sub\text{-}procesos_i\ adicionados) + \sum talla\ (sub\text{-}procesos_i\ modificados) + \sum (sub\text{-}procesos_i\ eliminados)$

Anexo C

IDENTIFICACIÓN DE REGLAS COSMIC-FFP

A fin de permitir referencias precisas, la siguiente tabla muestra los principios del método de COSMIC-FFP.

ID	NOMBRE	DESCRIPCIÓN
R-01	Capa	a) Los "paquetes" de servicio funcional, como los sistemas de gestión de bases de datos, las interfases gráficas del usuario, los sistemas operativos o los dispositivos pilotos son generalmente considerados como capas distintas. b) Si un software es concebido sobre la base de un paradigma arquitectural conocida, use el paradigma para identificar las capas. c) El nivel de la aplicación del software es generalmente considerado como la capa de más alto nivel. d) En caso de duda, usar los conceptos de cohesión para distinguir entre capas interactuantes.
R-02	Frontera	a) Empezar identificando los eventos desencadenadores, luego identifique los procesos funcionales desencadenados por esos eventos. La frontera se sitúa entre esos eventos y funciones. b) Identificar los dispositivos E/S utilizados por el software medido para interactuar con cada tipo de usuario. Relacionar los dispositivos E/S con las funciones provistas por el software medido. La frontera se sitúa entre dichas funciones y los dispositivos E/S. c) Para los software técnicos o en tiempo real, usar el concepto de capas para identificar la frontera (sección 3.1).
R-03	Proceso Funcional	a) Los sub-conjuntos de eventos desencadenadores no son considerados como eventos desencadenadores. Así, si una ocurrencia específica de un evento desencadena la entrada de un grupo de datos que comprende los atributos A, B y C, y que otra ocurrencia del mismo evento desencadena una entrada de un grupo de datos conteniendo valores de los atributos A y B solamente, este último no es considerado un evento desencadenador diferente. Él es considerado como el mismo desde el punto de vista de la identificación de procesos funcionales COSMIC-FFP. Consecuentemente, son identificados solamente una entrada y un proceso funcional, el cual manipula los atributos de datos A, B y C . b) En el contexto del software en tiempo real, un proceso funcional es igualmente desencadenado por un evento. Dicho proceso termina cuando un punto asincrónico de tiempo es alcanzado. Ese punto es alcanzado cuando, en una secuencia de movimientos de datos, uno de esos movimientos no está sincronizado con el precedente. Un punto de tiempo asincrónico es equivalente a un estado de espera auto-inducido.

ID	NOMBRE	DESCRIPCIÓN
R-04	Grupo de datos	a) APLICACIÓN AL SOFTWARE MIS. La práctica de medición ha establecido que, en el software MIS, un grupo de datos es identificado para cada tipo de entidad que se encuentra en el modelo de datos normalizado del software medido. Usualmente, dichos grupos de datos muestran una persistencia indefinida y el software es requerido para almacenar los datos de las entidades-tipos concernidas. Ejemplos: en el campo del software de información de gestión, un objeto podría ser "empleado" (físico) o "pedido" (conceptual) – el software es requerido para almacenar los datos de los empleados y los pedidos. Además, los grupos de datos que muestran una persistencia transitoria (i.e. grupos de datos sobre objetos transitorios) son formados donde haya una interrogación ad-hoc por datos sobre alguna cosa sobre la cual los datos no son almacenados con persistencia indefinida, pero que pueden ser derivados de datos almacenados con persistencia indefinida. En dichos casos el objeto transitorio es un movimiento de datos de entrada en la interrogación ad-hoc (la selección de datos para derivar los datos requeridos) y de los movimientos de datos de salida que contienen los atributos deseados del objeto transitorio . Ejemplo: supongamos una interrogación ad hoc a la base de datos de personal para extraer la lista de los nombres de todos los empleados mayores de 35 años. la entrada es un grupo de datos que contiene la selección de los parámetros. La salida es el grupo de datos que contiene la lista de los nombres. b) APLICACIÓN AL SOFTWARE EN TIEMPO REAL. La práctica de medición del software en tiempo real ha establecido que los grupos de datos en este tipo de software toman seguido las formas siguientes: • Movimientos de datos que son entradas de dispositivos físicos que típicamente contienen datos sobre el estado de un objeto (-tipo), o que pueden ser "el objeto" mismo . Ejemplo: en un intercambio de mensajes, el objeto puede ser un mensaje que requiere del software para su procesamiento, e.g. para enviar o almacenar y hacer seguir, directamente. • La estructura común de datos, representando objetos mencionados en los requerimientos funcionales del usuario, los que son conservados en memoria volátil y accesibles a la mayor parte de procesos del software medido. • La estructura de datos de referencia, representando gráficos o tablas de valores del FUR, los cuales son conservados en memoria permanente (i.e. ROM) y accesibles a la mayor parte de procesos del software medido, • Almacenamientos simples, representando objetos de los requerimientos funcionales del usuario que son conservados en dispositivos de memoria permanente.

ID	NOMBRE	DESCRIPCIÓN
R-05	ENTRADA	a) Los eventos desencadenados por temporalmente son considerados externos. Así, un evento que ocurre cada 3 segundos es asociado a una ENTRADA que desplaza un atributo de dato, por ejemplo. Sin embargo, el proceso funcional que genera el evento periódicamente es ignorado ya que ocurre, por definición, al exterior de la frontera del software. b) A menos que un proceso funcional específico sea necesario, la obtención del tiempo a partir de un sistema de reloj no es considerado como una ENTRADA. Por ejemplo, cuando un proceso funcional imprime automáticamente la hora, ninguna ENTRADA es identificada por la obtención del valor del tiempo provisto por el reloj interno.
R-06	SALIDA	a) Todos los mensajes generados sin datos del usuario (i.e. mensajes de confirmación y error) son identificados como una sola SALIDA en el proceso funcional dentro del cual esa SALIDA es identificada. Por ejemplo, supongamos los procesos funcionales A y B identificados en la misma capa. "A" puede emitir 2 m2nsajes de confirmación y 5 de error a los usuarios, y "B", 8 mensajes de error. En este ejemplo, serían identificadas: una SALIDA en el proceso "A" (manipulando 5+2=7 mensajes), y una SALIDA separada en el proceso "B" (manipulando 8 mensajes). .
R-07	Agregación de los resultados de la medición	a) Cuando la matriz del anexo A es utilizada, la agregación de los resultados puede ser realizada contando en número de sub-procesos en cada fila de una capa dada.

Anexo D

INFORMACIÓN ADICIONAL SOBRE LAS CAPAS DE SOFTWARE

El concepto de capas ha evolucionado a partir de las dificultades encontradas en la práctica[28] en la aplicación de las "reglas de frontera" del IFPUG Counting Practices Manual (up to v. 4.0). En las prácticas tradicionales de medición de talla funcional, la frontera del software es definida como el límite entre el software a ser medido y el usuario. Además, esta definición establece que la posición de la frontera es dependiente de la vista externa del usuario de la aplicación e independiente de consideraciones técnicas y/o de implementación.

Algunas preguntas surgen de esta definición:

- **Quién es el usuario ?**

  El método de medición COSMIC-FFP define al usuario como "un ser humano o dispositivos de software o de ingeniería que interactúan con el software medido".

- **Cómo se puede determinar cuál es la perspectiva del usuario ?**

  El enfoque del usuario, tal cual venimos de definirlo, implica varios niveles de abstracción. Al más alto nivel, una solución es declarada en términos amplios usando el lenguaje del medio ambiente del problema. A niveles más bajos, una orientación más procedural es utilizada. Cada etapa del proceso de ingeniería de software es un refinamiento en el nivel de abstracción de la solución software (Pressman97).

- **Cuándo se debe considerar una solución como técnica ?**

  El software que es usualmente incluido en el ámbito de una organización puede ser categorizado sobre la base de los tipos de servicios provistos, tal que ilustrado en la Figura D.1.

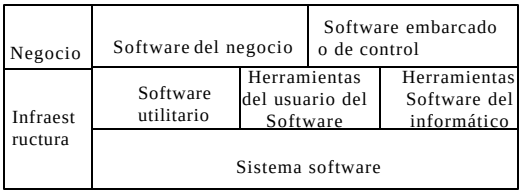


Figura D.1 – Categorías del software según el servicio provisto

Los tipos de servicio provistos serán definidos a continuación.

EJEMPLO – Funcionalidades provistas por diferentes tipos de software

Software del negocio
Este tipo de software provee funcionalidades que soportan el negocio principal de la organización. Los usuarios son primariamente usuarios humanos de negocios, sin embargo, una pequeña parte de las funcionalidades son provistas a, o provocadas por otras aplicaciones. Este tipo de software es típicamente comercial o de negocios (MIS) e incluye las aplicaciones de paga, cuentas por cobrar o sistemas de aprovisionamiento.

[28] Ver, por ejemplo: "Function Point Counting Practices for Highly Constrained Systems", Release 1.0, March 1993 by the European Function Point Users Group or "Comments on ISO 14143-Part 5", release V1b by Grant Rule.

EJEMPLO – Funcionalidades provistas por diferentes tipos de software

Software embarcado o de control
Este tipo de software provee igualmente funcionalidades que soportan los negocios o productos principales de la organización. Los usuarios son primariamente otras aplicaciones software embarcadas en equipos. Este tipo de software típicamente opera bajo estrictas condiciones de tiempo y se refiere seguido al software en tiempo real. Ejemplos de este tipo de software son los sistemas de monitoreo de equipos y los sistemas de intercambio de mensajes telefónicos.

Software utilitario
Este tipo de software provee funcionalidades para soportar el software de los negocios. Los usuarios son las aplicaciones de negocios mismas que demandan la operación de los utilitarios, pero pueden incluir informáticas, gestionarios o usuarios administrativos. Ejemplos de estos sistemas serían los utilitarios de backup (para asegurar la fiabilidad de los datos de las aplicaciones de negocios) y de almacenamiento (para optimizar la performance de las aplicaciones de negocios). Otros ejemplos son los software de instalación y de conversión.

Herramientas software del usuario
Este tipo de software provee herramientas funcionales utilizadas por los gestionarios para crear las funcionalidades de los software de negocios, Los usuarios primarios son los software de negocio mismos. Los usuarios humanos administrativos de estas herramientas pueden provenir de los sectores de negocios o de la informática de la organización. Ejemplos de este software serían los generadores de informes, las hojas de cálculo y los procesadores de texto.

Herramientas software para los informáticos
Estas herramientas son usadas para desarrollar las funcionalidades de los software de negocios. Los usuarios son primariamente otras aplicaciones, que son generadas, o usadas como input de otras aplicaciones. Los usuarios humanos son los informáticos que desarrollan sistemas o usuarios administrativos. Ejemplos de este tipo son los generadores de código, los software de test, los generadores de nuevos productos, etc.

Sistemas Software
Este tipo permite a todos lo otros software de operar y de proveer sus propias funcionalidades. Los usuarios son primariamente otras aplicaciones con una interfase limitada con el personal informático. Ejemplos son los sistemas operativos, los pilotos de impresión, los convertidores de protocolo y los software de presentación.

(Extraído de P. Morris et J.M. Desharnais, ver referencias)

Este enfoque descriptivo es conveniente para distinguir diferentes tipos de funcionalidad. Sin embargo, el conlleva algunas zonas grises cuando es usada para medir la talla funcional del software. El concepto de capas es propuesto como una manera de formalizar los diferentes tipos de funcionalidad con el propósito de medir la talla funcional del software.

A un nivel de servicio dado, el software provee funcionalidades a una categoría específica de usuarios. Cuando algunos ítems de un software proveen funcionalidades jerárquicamente relacionadas unas con otras, ellas son identificadas como perteneciendo a capas diferentes. Cada capa es entonces el "usuario" de la capa de inferior jerarquía. Las capas de software representan diferentes niveles de abstracción. Al más alto nivel de abstracción, una solución es establecida en términos generales usando el lenguaje del medio ambiente del problema. Al más bajo nivel de abstracción, una orientación más procedural es ofrecida. Una capa de nivel más alto está así en un nivel de abstracción superior que el de la capa más baja.

En la figura D2, la capa superior (L_1) es la que libra las funcionalidades al usuario final. El software en alguna de las capas puede también proveer funcionalidades a sistemas dentro de la misma capa.

Capa 1 (L_1), provee funcionalidades directas a los usuarios. Los usuarios finales son humanos en el casos de las aplicaciones de negocios, o equipos en el caso do software en tiempo real.

- Capa 2 (L_2), reúne los software de infraestructura que provee funcionalidades de soporte sus usuarios: software en L_1.
- Capa n, reúne los software que proveen soporte a sus usuarios: software en L_{n-1}.

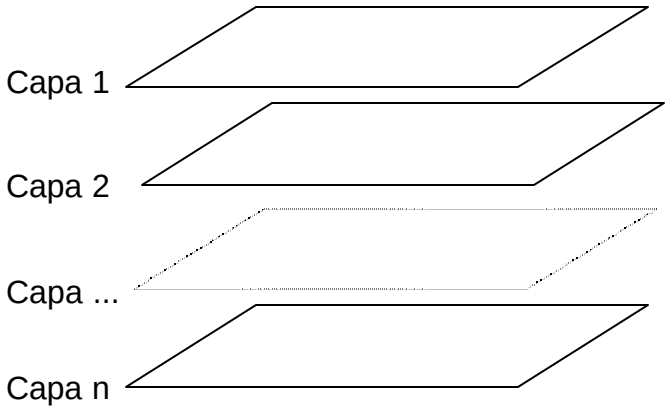


Figure E.2 – Representación genérica de las capas en un producto software

- **Cómo hacer una distinción entre diferentes capas**

Los conceptos de arquitectura de software son utilizados en ingeniería de software para comprender el conjunto de la estructura del software y las maneras en las que esa arquitectura provee de integridad conceptual al software (Pressman). Un arquitecto de software organiza una solución software para obtener el mejor conjunto de módulos. La modularidad efectiva puede ser lograda definiendo un conjunto de módulos independientes que sólo comunica don otro módulo la información necesaria para el logro de la función requerida (principio de la información oculta o *information hiding*).

El principio de la información oculta sugiere que los módulos son caracterizados por las decisiones de concepción que oculta cada uno de los otros. En otros términos, los módulos deben ser especificados y concebidos para que la información (procedimientos y datos) contenida dentro de un módulo sea inaccesible a otros módulos que no tienen necesidad de dicha información (Pressman).

La estructura del programa debe ser particionada horizontal y verticalmente. El enfoque más simple para la partición horizontal define tres partes: input, transformación de datos y output. La partición vertical sugiere que el control (toma de decisiones) y el trabajo deben ser distribuidos de arriba hacia abajo en la arquitectura del programa (Pressman).

COSMIC-FFP provee reglas para medir la partición horizontal (medida de proceso en términos de input, transformación de datos y output). La partición vertical sugiere que un proceso puede acceder a diferentes niveles o capas de software. De una capa a otra, las piezas de software son procesos independientes, inaccesibles a otros procesos que no tienen necesidad de esa información.

El concepto de independencia funcional es la ampliación de la modularidad y del concepto de información oculta. La independencia es medida utilizando dos criterios cualitativos: cohesión y acoplamiento. La cohesión es la medida de la fortaleza funcional relativa del módulo. El acoplamiento es la medida de la interdependencia relativa entre los módulos de software.

Anexo E

SUB-VERSIONES COSMIC-FFP

Esta anexo contiene el material adicional de las sub-versiones del método.

Referencia en la versión 2.0	Naturaleza del cambio	Comentarios
Reconocimientos Control de versión Prefacio	Editorial	Modificado para la emisión de la versión 2.1
Glosario	Técnica Editorial Editorial	• Definición de proceso funcional: Exclusión en la primera frase de "y ordenado", remplazo de la segunda frase por "desencadenado directa, o indirectamente via un "actor", por un Evento (-tipo), y es completado cuando se ejecuta todo lo que es requerido de ser hecho en respuesta a Evento desencadenador (-tipo) ". • Definición de requerimientos funcionales del usuario (FUR): adición de "sobre la base de la perspectiva del usuario" al final de la primera frase. • Definición de sub-proceso: al final del párrafo, inclusión de : "Un sub-proceso COSMIC- FFP equivale a la un tipo componente de base funcional. Un sub-proceso COSMIC-FFP expresa sólamente los requerimientos funcionales del usuario y excluye los requerimientos técnicos y de calidad".
Capítulo 2	Editorial Editorial Técnica	• Sección 2.1, 1er párrafo: incluír al final de la 2da frase: "desde la perspectiva del usuario". • Sección 2.3, 1er párrafo: incluír después de la 1ra frase: "El modelo de software generado corresponde al sub-conjunto de los FURs a ser incluídos en el ámbito específico de la medición de la talla funcional". • Sección 2.3.2: 2do párrafo modificado para reflejar las modificaciones en la definición de proceso funcional. • Sección 2.4: al final de la frase, adición de "o una escritura".
Capítulo 3	Editorial Editorial Técnica	• Correcciones a la figura 3.1: texto editado y modificaciones en las lineas entre los rectángulos. • Sección 3.1, remplazo del 1er párafo por "Los requerimientos funcionales del usuario pueden explicitar, implicar o el analista de medición puede inferir, que los FURs se aplican al software en diferentes capas o items comparables, para los que la talla debe ser medida separadamente. Alternativamente, el analista de medición puede encontrarse con que el software aparece en diferentes capas o items comparables. En ambos casos, se necesitará de directivas que ayuden a decidir si los FURs o el software comprende una o más capas o items comparables. Las capas pueden ser identificadas según la definición y principios siguientes". • Sección 3.1, caja entitulada "Principios – Capa": remplazo del contenido por los acápites siguientes: a) El software en todas las capas provée funcionalidades a los usuarios (Un usuario puede ser un ser humano, un dispositivo físico u otro software). b) El software en una capa subordinada provee servicios funcionales al software en las capas clientes. c) El software en las capas subordinadas puede ejecutarse sin la asistencia del software en una capa cliente. d) El software en una capa cliente puede no ejecutarse apropiadamente si

		el software en la capa subordinada que depende no es ejecutado apropiadamente. e) El software en la capa cliente no usa necesariamente todas las funcionalidades provistas por el software en una capa subordinada. f) El software en una capa subordinada puede ser una capa cliente desde la perspectiva de una tercera capa subordinada. g) El software en las capas clientes y subordinadas puede físicamente compartir e intercambiar datos. Sin embargo, el software en cada capa interpreta los datos diferentemente. h) El software que comparte datos con otro software no puede ser considerado en una capa diferente si interpreta los datos de manera idéntica al otro software.
	Editorial	• Sección 3.1, parte encabezada por "Identificación de las fronteras de las capas": exclusión de este encabezamiento con todo lo que sigue hasta el final de 3.1 (incluída la figura 2.1.1).
	Editorial	• Seción 3.2, 1er párafo, adición de "Una vez que la frontera ha sido indentificada, cada FUR describiendo procesos al interior de la frontera es incluído dentro del ámbito de la medición de la talla funcional".
	Técnica	• Seción 3.3, 1er párrafo: adición de : "El ámbito de los FURs a ser medidos corresponde a todos los FURs que describen los procesos funcionales identificados". • Sección 3.3, Definición de proceso funcional: misma modificación que la señalada lineas arriba, en la sección Glosario.
Capítulo 4	Editorial	• Después del 1er párrafo, adición de : "La derivación de la talla funcional del software a ser medido es independiente del esfuerzo requerido para desarrollarlo o mantenerlo, del método usado para desarrollarlo o mantenerlo y de cualquier componente físico o tecnológico del software".
	Editorial	• Sección 4.1, 1er párrafo: después del párrafo, adición de "Una vez que los sub-procesos sean identificados, sus tipos serán identificados de acuerdo a los procedimientos definidos en las páginas siguientes ".
	Editorial	• Sección 4.1, 1era definición (sub-proceso): al final del párrafo, adición de "Un sub-proceso COSMIC-FFP expresa sólamente los requerimientos funcionales del utilizador y excluye los requerimientos técnicos y de calidad".
	Editorial	• Sección 4.2, último párrafo: exclusión de la referencia a las pruebas de campo. Las mediciones llevadas a cabo en las pruebas de campo COSMIC-FFP produjeron datos de software en tiempo real que soportan las hipótesis de dar a todos los movmientos de datos (Entradas, Salidas, Lecturas y Escrituras) la misma talla de un Cfsu. Los usuarios del método COSMIC-FFP son alentados a colectar y publicar informes con datos adicionales. El equipo COSMIC se reserva el derecho de cambiar o refinar la unidad standard de medida a la luz de los datos futuros. Los Usuarios del método COSMIC-FFP son, por lo tanto, alentados a recoger datos y contar separadamente las Entradas, Salidas, Lecturas y Escrituras lo que debería ser beneficioso para un posterior refinamiento de la unidad de medida.
Capítulo 5	Editorial	Cambio de convención para la representación de la unidad de talla, de "CFSU" a "Csfu". Realizado a través del texto.
Anexo B	Técnica	En la sección "nombre = capa", remplazo del contenido de "DESCRIPCIÓN" por el mismo contenido de 3.1 remplazado más arriba.
Otros	Editorial	• Los pie de página a través del texto han sido actualizados con referencias a los documentos ISO. • Sección de Referencias: actualización de la lista de referencias.

Referencias

Abran, A., Symons, C., Oligny, S., 'An Overview of COSMIC-FFP Field Trial Results', 12th *European Software Control and Metrics Conference – ESCOM 2001*, April 2-4, London (England), 2001.

Abran, A., Symons, C., Desharnais, J.M., Fagg, P., Morris, P., Oligny, S., Onvlee, J., Meli, R., Nevalainen, R., Rule, G. and St-Pierre, D., "COSMIC FFP Field Trials Aims, Progress and Interim Findings," *11th European Software Control and Metric Conference - ESCOM SCOPE 2000*, Munich, Germany, 2000.

Abran, A., Desharnais,J.M., Oligny, S., St-Pierre, D. and Symons, C., "COSMIC FFP – Measurement Manual Version 2,0 - Field Trials Version" Université du Québec à Montréal, Montréal, October 1999.

Abran, A. "FFP Release 2,0: An Implementation of COSMIC Functional Size Measurement Concepts," *Federation of European Software Measurement Associations - FESMA'99*, Amsterdam, 1999.

Abran, A., "Functional Size of Real-Time Software: Challenges & Solution," Tokyo, Japan: *Japanese Function Point Users Group - JFPUG*, 1998.

Abran, A., St-Pierre, D., Maya, M. and Desharnais, J.M., "Full Function Points for Embedded and Real-Time Software," *United Kingdom Software Metrics Association - UKSMA*, London, UK, 1998.

Abran, A., Desharnais, J.M., Maya, M., St-Pierre, D., and Bourque, P., "Design of Functional Size Measurement for Real-Time Software," Université du Québec à Montréal, Technical Report no. 13, November 23, Montréal (Canada),1998.

Abran, A., Maya, M., Desharnais, J.M., and St-Pierre, D., "Adapting Function Points to Real-Time Software," *American Programmer*, Vol. 10, no 11, p. 32-43, 1997.

Albrecht A.J., Gaffney J.E. Jr., "*Software function, source lines of code and development effort prediction: a software science validation*", IEEE Transactions on Software Engineering, Vol. SE-9, pp. 639-648, November 1983.

Albrecht, A.J., *AD/M Productivity Measurement and Estimate Validation*, IBM Corporate Information Systems, IBM Corp., Purchase, N.Y., May 1984.

Bevo, V., Lévesque, G., and Abran, A., "Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendu des premiers essais d'application et questions," *International Workshop on Software Measurement - IWSM'99*, Lac Supérieur, Canada, 1999.

Bootsma, F., 'Applying Full Function Points to Drive Strategic Business Improvement within the Real-time Software Environment', *IFPUG Annual Conference*, New Orleans, Oct. 18-22, 1999.

Buerhen, G. , Koll, I., 'First Experiences with the Introduction of an Effort Estimation Process', *ICSSEQ 99*, Dec. 8-10, Paris, France, 1999.

Buerhen, G., Foltin, E., Weber, E. and Schweikl, S., 'Applicability of FFP at Siemens AT', *International Workshop on Software Measurement – IWSM 99*, Lac Supérieur Québec, Canada, 1999.

Cooling, J. E., *Software Design for Real-Time Systems,* Chapman and Hall, 1991.

Desharnais, J.M., Abran, A., Oligny, S., St-Pierre, D. and Symons, C., "COSMIC FFP – Manuel de mesures Version 2.0 - Essais sur le terrain", Université du Québec à Montréal, Montréal, October 2000.

Desharnais, J.M., St-Pierre, D., Oligny, S. and Abran, A., "Software Layers and Measurement," *International Workshop on Software Measurement – IWSM,* Lac Supérieur, Québec, 1999.

Desharnais, J.M., Abran, A. and St-Pierre, D., "Functional Size of Real-Time Software," *11th International Conference - Software Engineering and its Applications*, Paris, France, 1998.

Foltin, E. 'Die Full-Function-Point-Methode', *Ein Ansatz zur Bestimmung des funktionalen Umfangs von Real-Time-Software*, Austria, 2000.

Ho, V.T., Abran, A. and Oligny, S., "Using COSMIC-FFP to Quantify Functional Reuse in Software Development," *11th European Software Control and Metric Conference - ESCOM SCOPE 2000,*. Munich, Germany, 2000.

Ho, V.T., Abran, A. and Fournier, B., "On the Reproducibility of a Functional Size Measurement Method and its Use for Determining the Range of Errors in the Measurement of a Particular Software Portfolio," Université du Québec à Montréal - UQAM, January 27, 2000.

Ho, V.T., Abran, A. and Oligny, S., "Case Study - Rice Cooker, Version 1.0," Université du Québec à Montréal, Montréal, Québec - UQAM, December 1999.

Ho, V.T.,. Abran, A., Oligny, S., Bevo, B., Ndiaye, I., Lakrib, C., Hoyos, M., Boudaa, A., Dion, D., Mattalah, M. and Dinh, H.P., "Case Study - ISND Loop Back Tester, Version 1.0," Université du Québec à Montréal, Montréal - UQAM, December 1999.

ISO, International Vocabulary of Basic and General Terms in Metrology", International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1.

ISO, ISO/IEC 14143-1998 – Software Engineering – Software measurement – Functional size measurement – Part 1 : Definition of concepts.

Kececi, N., LI, M. and Smidts, C., 'Function Point Analysis: An Application to a Nuclear Reactor Protection System', *Probabilistic Safety Assessment – PSA'99*, August 22-25, Washington, 1999.

Laplante, P., *Real-Time Systems Design and Analysis: An Engineer's Handbook*, The Institute of Electrical and Electrical Engineers Inc., New York, NY, 1993, 339 pages.

Lam M., Couzens G., Goodman P., Laing B., Rule G., *Function Point Counting Practices for Highly Constrained Systems*, European Function Point Users Group, 1993, 36 pages.

Le Petit Larousse Illustré, 1996 Edition.

Lokan, C. and Abran, A., "Multiple viewpoints in functional size measurement," *International Workshop on Software Measurement – IWSM 99*, Lac Supérieur, Québec, 1999.

Maya, M., Abran, A., Oligny, S., St-Pierre, D. and Desharnais, J.M., "Measuring the Functional Size of Real-Time Software," *9th European Software Control and Metrics Conference and 5th Conference for the European Network of Clubs for Reliability and Safety of Software - ESCOM-ENCRESS-98*, Rome, Italy, 1998.

Meli, R., Abran, A., Ho, V.T. and Oligny, S., "On the Applicability of COSMIC-FFP for Measuring Software Throughout its Life Cycle", *11th European Software Control and Metric Conference - ESCOM SCOPE 2000*, Munich, Germany, 2000.

Molinié, L.c Miranda, M., Abran, A., Desharnais, J.M., Oligny, S.,. St-Pierre, D. and Symons, C., "COSMIC FFP - Manual de Medición Versión 2.0 – "Versión de pruebas de campo"," Université du Québec à Montréal - UQAM, October 1999.

Morris, P. and Desharnais, J.M. 'Measuring All the Software not Just What the Business Uses', *IFPUG Fall Conference*, Orlando, Florida, Sept. 21-25, 1998.

Nevalainen, R., "COSMIC - taivaan lahja ohjelmiston koon laskentaan," *Swengineering - Project Special*, Finland, vol. 1, pp. 6-7, 2001.

Oligny S. and Abran, A., "On the Compatibility Between Full Function Points and IFPUG Function Points," *10th European Software Control and Metric Conference - ESCOM SCOPE 99,* Herstmonceux Castle, England, 1999.

Oligny, S., Abran, A. and St-Pierre, D., "Improving Software Functional Size Measurement," *COCOMO and Software Cost Modeling International Forum 14*, Los Angeles, USA, 1999.

Oligny, S., Abran, A., St-Pierre, D. and Desharnais, J.M., "Innovations dans la mesure de la taille fonctionnelle du logiciel," *12th International Conference on Software and Systems Engineering and their Applications - ICSSEA*, Paris, France, 1999.

Oligny, S., Abran, A., Desharnais, J.M., St-Pierre, D. and Symons, C., "COSMIC-FFP Technology Transitioning Data Collection Protocol," Université du Québec à Montréal, Montréal – UQAM, Version 1c, December 1999.

Oligny, S., Desharnais, J.M. and Abran, A., "A Method for Measuring the Functional Size of Embedded Software," *3rd International Conference on Industrial Automation*. Montréal, 1999.

Oligny S. and Abran, A., "Field Testing Full Function Points: Recent Results," presented at *NESMA Autumn Congress 1998*, Amsterdam, Netherlands, 1998.

Oligny S. and Meli, R. "COSMIC FFP - Aims, Design Principles and Progress", GUFPI – ISMA (Gruppo Utenti Function Point Italia – Italian Software Metrics Association), Rome, November 1999

Pressman, R., *Software Engineering – A practitioner's approach*, 4th Edition, McGraw Hill, 1997.

Rules G., *Comments on ISO 14143 Part 5*, Release V1b.

St-Pierre, D., Maya, M., Abran, A., Desharnais, J.M., and Bourque, P., "Full Function Points: Counting Practices Manual," Université du Québec à Montréal, Montréal, Technical 1997-04, September 1997.

St-Pierre, D., Abran, A., Araki, M. and Desharnais, J.M., "Adapting Function Points to Real-Time Software," *IFPUG 1997 Fall Conference*, Scottsdale, Arizona,1997.

St-Pierre, D., Desharnais, J.M., Abran, A. and Gardner, B., "Definition of When Requirements Should be Used to Count Function Points in a Project," in *The Voice 1996 - A publication of the International Function Point Users Group*, Vol. 1, no 1, 1996, p. 6-7, 22.

Symons, C., Abran, A., Desharnais, J.M., Fagg, P., Goodman, P., Morris, P., Oligny, S., Onvlee, J., Nevalainen, R., Rule, G. and St-Pierre, D., "COSMIC FFP - Buts, approche conceptuelle et progrès," presented at *ASSEMI*, Paris, France, 1999.

Symons, C., Abran, A., Dekkers, C., Desharnais, M.M., Fagg, P. Morris, P., Onvlee, J., Nevalainen, R., Rule, G. and St Pierre, D., "Introduction and Overview of Principles," *Japanese Function Point User Group - JFPUG*, Tokyo, Japan, 1999.