



[

**El Método COSMIC del Tamaño Funcional del Software
Versión 4.0.1**

Manual de Medición

(Guía COSMIC de implementación de ISO/IEC 19761: 2011)

Julio 2015

Agradecimientos

Version 4.0 and 4.0.1 Revisores		
Alain Abran École de Technologie Supérieure, Université du Québec Canada	Diana Baklizky TI Metrics Brazil	Jean-Marc Desharnais École de Technologie Supérieure – Université du Québec Canada
Peter Fagg Pentad United Kingdom	Cigdem Gencel Free University of Bolzano-Bozen Italy	Charles Symons* United Kingdom
Jayakumar K. R. Amitysoft India	Arlan Lesterhuis* The Netherlands	Bernard Londeix Telmaco United Kingdom
Shin-Ichi Nagano NTT Comware Japan	Luca Santillo Agile Metrics Italy	Hassan Soubra Ecole Supérieure des Techniques Aéronautiques et de Construction Automobile France
Sylvie Trudel Université du Québec à Montréal Canada	Monica Villavicencio ESPOL Ecuador	Frank Vogelesang Ordina The Netherlands
Chris Woodward Chris Woodward Associates Ltd. United Kingdom		

* Editores de la versión 4.0.1 del método COSMIC

Para los revisores de las versiones anteriores del método COSMIC, consulte esos documentos.

Equipo de traducción de la Versión 4.0.1 de COSMIC al Español		
Francisco Valdés Souto SPINGERE Mexican Software Metrics Association (AMMS)	Mauricio Aguiar TI Metrics	

Derechos de Autor 2015. Todos los derechos reservados al The Common Software Measurement International Consortium (COSMIC). Permiso para copiar la totalidad o parte de este material se concede a condición de que las copias no se hacen ni distribuyen para ventaja comercial y que el título de la publicación, su número de versión, y su fecha se citan y se da aviso que la copia es con permiso del Common Software Measurement International Consortium (COSMIC). Para copiar de otro modo requiere permiso específico

Una versión de dominio público del Manual de Medición de COSMIC y otros informes técnicos, incluyendo traducciones a otros idiomas se puede encontrar en el portal de la www.cosmic-sizing.org.

Control de Versiones

El siguiente cuadro muestra la historia de las versiones de este document.

FECHA	REVISOR(ES)	Modificaciones / Adiciones
1999-03-31	Serge Oligny	Primer borrador, publicado para comentarios a los colaboradores.
1999-10-29	Equipo Principal COSMIC	Revisado, comentarios finales antes de la publicación de "prueba de campo" versión 2.0.
2001-05-01	Equipo Principal COSMIC	Revisado para la conformidad con la norma ISO / IEC 14143-1: 1998 + aclaraciones sobre las reglas de medición para la versión 2.1.
2003-01-31	Comité de Prácticas de Medición (Measurement Practices Committee) COSMIC	Revisado para la conformidad con la norma ISO / IEC FDIS 19761: 2002 + más aclaraciones sobre las reglas de medición para la versión 2.2.
2007-09-01	Comité de Prácticas de Medición (Measurement Practices Committee) COSMIC	Revisada para más aclaraciones y adiciones a las reglas de medición para la versión 3.0, particularmente en el área de la fase de estrategia de medición. El nombre del método fue cambiado del "método "COSMIC-FFP" al de "método COSMIC". En la actualización a la versión 3.0 de v2.2, distintas partes del "Manual de Medición" v2.2 fueron separadas en otros documentos.
2009-05-01	Comité de Prácticas de Medición (Measurement Practices Committee) COSMIC	Versión 3.0 revisada para v3.0.1 para hacer mejoras de redacción y aclaraciones, y distinguir ejemplos con mayor claridad. Esta versión también incorpora los cambios propuestos en los Boletines de Actualización del Método (Method Update Bulletins) 3, 4 y 5.
2014-04-01	Comité de Prácticas de Medición (Measurement Practices Committee) COSMIC et al	Versión 4.0 revisada para tener en cuenta la actualización de los Boletines de Actualización del Método (Method Update Bulletins) 6 a 11, varias mejoras de redacción y la incorporación del Glosario de términos. Véase el Apéndice E para obtener detalles de estos cambios.
2015-04-01	Comité de Prácticas de Medición (Measurement Practices Committee) COSMIC et al	Versión 4.0.1 revisada para tener en cuenta un error en la versión v4.0 y varias mejoras de edición. Ver el Apéndice E para detalle de estos cambios.

El método COSMIC ofrece un método estandarizado de medición de un tamaño funcional del software de los dominios conocidos comúnmente como 'aplicación de negocios' (o 'MIS') de software, software en 'tiempo real', software de 'infraestructura' y algunos tipos de científicos / software de ingeniería.

El método COSMIC fue inicialmente aceptada por la norma ISO / IEC JTC1 SC7 como Norma Internacional en Diciembre de 2002. La versión actual es la norma ISO / IEC 19761: 2011 'Software Engineering – COSMIC – A functional size measurement method' [1] (en lo sucesivo, el estándar ISO/IEC 19761).

El estándar ISO/IEC 19761 sólo contiene las definiciones normativas fundamentales y las reglas del método como en la versión 3.0 del método. El propósito del Manual de Medición es proporcionar estas reglas y definiciones, y también proporcionar una mayor explicación y muchos más ejemplos para ayudar Medidores a entender completamente cómo aplicar el método. El Manual de Medición es la descripción estándar principal del método COSMIC para uso práctico.

Introducción al método COSMIC.

Además del 'Manual de Medición', la 'Introducción al método COSMIC de software de medición' [2] da un resumen de la versión 4.0 / 4.0.1 del método¹. Este documento de 'Introducción' debe leerse primero por cualquier persona que es nueva en la medición del tamaño funcional ('FSM') o que esté familiarizado con otro método FSM y está pensando en convertir, o que simplemente quiere una visión general del método COSMIC, antes de la lectura este Manual de Medición. Mucha información de antecedentes sobre FSM y el método COSMIC, así como directrices de apoyo (por ejemplo, sobre la manera de aplicar el método en diversas circunstancias especiales, etc.), estudios de casos, trabajos de investigación, etc., se pueden encontrar en el portal de la www.cosmicsizing.org.

Principales cambios para la versión 4.0.1 del Método COSMIC

El cambio en la denominación de esta versión 4.0.1 del método COSMIC desde el anterior 4,0 indica que esta versión se ocupa principalmente de hacer algunas correcciones de redacción en v4.0 del Manual de Medición. Esta v4.0.1 no introduce cambios en los principios² básicos del método. Los únicos cambios que afectan a las definiciones y las reglas son las siguientes.

La definición de 'Requerimiento no funcional' se ha hecho más precisa y ahora excluye los requisitos del proyecto.

Las definiciones de 'objeto de interés' y 'proceso funcional' se han aclarado aún más.

Las definiciones de un 'movimiento de datos, y de una Entrada, Salida, Lectura y Escritura se han editado para describir con más precisión cómo se *consideran* para (no incluye) la manipulación de datos asociada.

Las reglas para la 'singularidad de un movimiento de datos y las posibles excepciones' han sido reformuladas y un error se ha corregido.

La definición de 'mensaje / confirmación de error' se ha clarificado. Una nueva regla se ha añadido para hacer las reglas de agregación tamaño más clara en el caso de los mensajes de error.

Las razones de la mayoría de los cambios fueron publicados en una "Lista de fe de erratas y correcciones" en agosto de 2014.

Todos los principales cambios se resumen en el Apéndice E. COSMIC prevé que las mejoras introducidas en la versión 4.0 y en esta versión 4.0.1 del método se someterán a ISO para su inclusión en la norma ISO / IEC 19761, cuando le toque la siguiente revisión.

¹ La publicación de la version 4.0 y 4.0.1 del manual de Medición incorporan el Glosario de Términos, lo que significa que el documento de la v3.0.1 denominado 'Documentación y Glosario de Términos', es obsoleto ahora. También el document de la v3.0 denominado 'Advanced & Related Topics' será remplazado por dos guías que son 'Guía para la aproximación de la medición de tamaño funcional COSMIC' [6] y 'Guía sobre Convertibilidad' [13] que actualmente están en desarrollo.

² Dos de los principios para el grupo de datos se han eliminado, en virtud de que no agregan información útil.

Este Manual de Medición para la versión 4.0.1 del método se convierte en la definición estándar actual del método desde abril de 2015. El Comité de Prácticas de Medición (MPC) ha trabajado duro para eliminar los defectos y mejorar el documento para que sea más fácil de entender pero es posible que defectos permanecen y/o que cierto texto siga siendo difícil de entender. Instamos encarecidamente a los lectores y traductores a contactar al MPC reportar cualquier defecto y/o texto que no esté claro, utilizando el proceso del Apéndice G.

Consecuencias de los principales cambios a v4.0.1 en mediciones de tamaño existentes, etc.

Los principios básicos originales del método COSMIC se han mantenido sin cambios desde que fueron publicadas por primera vez en el primer borrador del Manual de Medición en 1999. Esto es incluso a pesar de las diversas mejoras y adiciones necesarias para producir el Estándar Internacional y para producir todas las versiones del método hasta esta última versión 4.0.1.

Tamaños funcionales medidos de acuerdo con los principios y normas de la versión 4.0.1 del Manual de Medición pueden diferir de los tamaños medidos utilizando versiones anteriores sólo porque las nuevas reglas tienen la intención de ser más precisas y completas. De ahí que Medidores tienen menos discrecionalidad para la interpretación personal de las reglas que era posible con versiones anteriores.

Como indicación adicional de la continuidad del método, los que pasaron el examen de certificación de nivel de la Fundamentos para la versión 3.0 / 3.0.1 / 4.0 del método serán considerados para ser aún certificada para v4.0.1 del método en el nivel de Fundamentos.

Nota sobre la terminología

Para la terminología utilizada en este Manual de Medición, consulte el Glosario en el Apéndice F. Este Manual de Medición utiliza terminología estándar ISO, es decir,

'deberá' indica una regla es obligatorio; 'debería' indica una regla es consultivo. (Si ningún término está presente asumir 'deberá'.)

'podría' indica 'se permite'; 'puede' indica 'es capaz de'

El contenido de esta Guía

Capítulo 1 trata los tipos de software para las que puede utilizarse el método COSMIC. El término 'Requisitos Funcionales de Usuario' ('FUR'), se define, junto con los principios básicos del método COSMIC. El proceso de medición método COSMIC y la unidad de medida también se definen.

Capítulo 2 describe la primera fase de Estrategia de Medición del proceso de medición en términos de sus parámetros clave, como el propósito de la medición, el alcance de la medición y los usuarios funcionales de una pieza de software. Estos parámetros deben ser definidos antes de comenzar a medir así el significado de las mediciones resultantes puede ser entendido y convenido.

Capítulo 3 se analiza la segunda fase de Mapeo del proceso de medición mediante la definición de cómo los FUR deben representarse con los procesos funcionales y sus movimientos de datos. Un movimiento de datos mueve un grupo de datos, que consiste en atributos de datos que describen todos ellos un 'objeto de interés'. Se definen los cuatro tipos de movimientos de datos: Las Entradas mueven datos desde y las Salidas mueven los datos hacia usuarios funcionales respectivamente; Lecturas mueven datos desde y Escritura mueven datos al almacenamiento persistente respectivamente.

Capítulo 4 describe la fase de Medición final del proceso de medición. Se definen las reglas para la asignación de un tamaño a los FUR de una pieza de software y la forma de agregar diferentes tamaños de piezas de software. Este capítulo también trata cómo medir los cambios al software y analiza la posibilidad de realizar 'extensiones locales' al método COSMIC estándar.

Capítulo 5 se indican los parámetros que se deben considerar para registrar las mediciones.

Los Apéndices A al E proporcionan un poco más de detalle, un resumen de los principios del método, de las reglas y los principales cambios a este Manual de Medición desde la v3.0.1 a la v4,0 y de la v4,0 a la v4.0.1.

Apéndice F contiene el Glosario de términos del método.

El Common Software Measurement International Consortium (COSMIC)

COSMIC es una organización voluntaria, sin fines de lucro, de expertos de métricas de software de todo el mundo, fundada en 1998. Todas sus publicaciones son 'abiertas' y disponibles para su distribución gratuita sujeta a las restricciones de derechos de autor y de acuse de recibo. Para más información sobre COSMIC y su organización, consulte el sitio web de COSMIC www.cosmic-sizing.org.

Comité de Prácticas de Medición COSMIC

Tabla de Contenido

1	INTRODUCTION.....	10
1.0	Resumen del Capítulo.....	10
1.1	Aplicabilidad del Método COSMIC.....	10
1.2	Requisitos Funcionales de Usuario.....	10
1.2.1	<i>Extracción de los requisitos funcionales de usuario a partir de los artefactos del software en la práctica.....</i>	<i>11</i>
1.2.2	<i>Extracción o deducción de los requisitos funcionales de usuario a partir de artefactos software.....</i>	<i>13</i>
1.2.3	<i>Requisitos NO-Funcionales (Non-Functional Requirements, NFR)</i>	<i>13</i>
1.3	Los Principios Fundamentales del Método COSMIC.....	14
1.3.1	<i>El Modelo Contextual de Software de COSMIC.....</i>	<i>15</i>
1.3.2	<i>El Modelo Genérico de Software</i>	<i>15</i>
1.3.3	<i>Tipos contra Ocurrencias.....</i>	<i>16</i>
1.4	El Proceso de Medición COSMIC y la Unidad de Medida	17
1.5	Limitaciones sobre la Aplicabilidad del Método COSMIC.....	18
2	LA FASE DE ESTRATEGIA DE MEDICIÓN	19
2.0	Resumen del Capítulo.....	19
2.1	Definir el propósito de la Medición	20
2.1.1	<i>El propósito de una medición determina el tamaño que será medido</i>	<i>21</i>
2.1.2	<i>La importancia del propósito.....</i>	<i>21</i>
2.2	Definir el alcance de la medición	22
2.2.1	<i>Derivando el alcance a partir del propósito de una medición.....</i>	<i>22</i>
2.2.2	<i>Capas.....</i>	<i>24</i>
2.2.3	<i>Niveles de descomposición</i>	<i>27</i>
2.2.4	<i>Definiendo el alcance de la medición: Resumen.....</i>	<i>27</i>
2.3	Identificando usuarios funcionales y el almacenamiento persistente	27
2.3.1	<i>El tamaño funcional puede variar con los usuarios funcionales.....</i>	<i>27</i>
2.3.2	<i>Almacén persistente</i>	<i>29</i>
2.3.3	<i>Diagramas de Contexto</i>	<i>30</i>
2.4	Identificando el nivel de granularidad.....	31
2.4.1	<i>La necesidad de un nivel de granularidad estándar.....</i>	<i>31</i>
2.4.2	<i>Aclaración del nivel de granularidad.....</i>	<i>32</i>
2.4.3	<i>El nivel de granularidad estándar</i>	<i>32</i>
2.5	Observaciones finales sobre la Fase de Estrategia de Medición	36
3	FASE DE REPRESENTACIÓN	37
3.0	Resumen del Capítulo.....	37
3.1	Representación de los FUR al Modelo Genérico de Software	37
3.2	Identificación de los procesos funcionales.....	39
3.2.1	<i>Definiciones</i>	<i>39</i>
3.2.2	<i>La aproximación para identificar procesos funcionales.....</i>	<i>41</i>
3.2.3	<i>Eventos disparadores y procesos funcionales en el ámbito de aplicaciones de negocio</i>	<i>42</i>
3.2.4	<i>Eventos disparadores y procesos funcionales en el ámbito de aplicaciones de tiempo real</i>	<i>44</i>
3.2.5	<i>Más sobre procesos funcionales separados</i>	<i>44</i>
3.2.6	<i>La medición de los componentes de un sistema de software distribuido</i>	<i>45</i>
3.2.7	<i>Independencia de los procesos funcionales compartiendo algunas funciones comunes o similares: Reutilización</i>	<i>46</i>
3.2.8	<i>Eventos que desencadenan la ejecución de un sistema de software</i>	<i>46</i>
3.3	Identificación objetos de interés y grupos de datos	47
3.3.1	<i>Definiciones y principios</i>	<i>47</i>

3.3.2	<i>Sobre la materialización de un grupo de datos</i>	48
3.3.3	<i>Sobre la identificación de objetos de interés y grupos de datos</i>	48
3.3.4	<i>Grupos de datos o datos que NO son candidatos a movimientos de datos</i>	49
3.3.5	<i>El usuario funcional como objeto de interés</i>	50
3.4	Identificación de los atributos de datos (opcional)	50
3.4.1	<i>Definición</i>	50
3.4.2	<i>Sobre la asociación de atributos de datos y grupos de datos</i>	50
3.5	Identificando los movimientos de datos	50
3.5.1	<i>Definición de los tipos de movimientos de datos</i>	51
3.5.2	<i>Identificando entradas (E)</i>	52
3.5.3	<i>Identificando de salidas (X)</i>	53
3.5.4	<i>Identificando Lecturas (R)</i>	54
3.5.5	<i>Identificando escrituras (W)</i>	55
3.5.6	<i>Sobre las manipulaciones de datos asociadas con los movimientos de datos</i>	55
3.5.7	<i>Movimientos de datos únicos y posibles excepciones</i>	57
3.5.8	<i>Cuando un proceso funcional mueve datos para o desde el almacenamiento persistente</i>	59
3.5.9	<i>Cuando un proceso funcional requiere datos de un usuario funcional</i>	63
3.5.10	<i>Comandos de navegación y de control de visualización para los usuarios humanos (Comandos de Control)</i>	65
3.5.11	<i>Mensajes de Error/Confirmación y otras indicaciones de condiciones de error</i>	66
4	LA FASE DE MEDICIÓN	68
4.0	Resumen del Capítulo	68
4.1	La fase de medición del proceso	68
4.2	Aplicando la unidad de medida COSMIC	68
4.3	Agregando los resultados de medición	69
4.3.1	<i>Reglas generales de agregación</i>	69
4.3.2	<i>Más sobre la agregación del tamaño funcional</i>	70
4.4	Más en la medición del tamaño de los cambios de software	70
4.4.1	<i>Modificando funcionalidades</i>	71
4.4.2	<i>Tamaño del software funcionalmente modificado</i>	73
4.5	Extendiendo el método de medición COSMIC	73
4.5.1	<i>Introducción</i>	73
4.5.2	<i>Manipulación de software rico en datos</i>	73
4.5.3	<i>Limitaciones de los factores que contribuyen al tamaño funcional</i>	74
4.5.4	<i>Limitaciones en la medición de piezas de software muy pequeñas</i>	74
4.5.5	<i>Extensiones locales con algoritmos complejos</i>	74
4.5.6	<i>Extensión Local con sub-unidades de medida</i>	74
5	INFORME DE MEDIDAS	76
5.0	Resumen del Capítulo	76
5.1	Etiquetado	76
5.2	Archivando los resultados de medición COSMIC	77
6	REFERENCES	78
APÉNDICE A – DOCUMENTACIÓN DE UNA MEDICIÓN DEL TAMAÑO COSMIC		79
APÉNDICE B – EVOLUCIÓN DE LOS REQUISITOS NO FUNCIONALES – EJEMPLOS		80
APÉNDICE C - CARDINALIDAD DE EVENTOS DESENCADENANTES, USUARIOS FUNCIONALES Y PROCESOS FUNCIONALES		81
APÉNDICE D – RESUMEN DE PRINCIPIOS Y REGLAS DEL MÉTODO COSMIC		84
APÉNDICE E – LOS PRINCIPALES CAMBIOS DE VERSIÓN 3.0.1 A LAS VERSIONES 4.0 Y V4.0.1		95
<i>E1: Principales cambios respecto de v3.0.1 a v4.0</i>		95

<i>E2: Principales cambios de v4.0 a v4.0.1</i>	<i>97</i>
APÉNDICE F – GLOSARIO DE TÉRMINOS	99
APÉNDICE G - CAMBIO DE SOLICITUD Y PROCEDIMIENTO DE COMENTARIO	105

INTRODUCTION

1.0 Resumen del Capítulo

Este capítulo tiene cuatro propósitos:

Explicar los tipos de software para los que se pueden utilizar el método COSMIC ("ámbitos aplicables), y las limitaciones en su uso.

Para definir 'Requisitos Funcionales de Usuario' (FUR), es decir, los requisitos de la funcionalidad del software que el método COSMIC pretende medir. Explicamos en términos generales cómo un medidor puede extraer o derivar el FUR de artefactos de software disponibles para una medición de tamaño funcional. Los Requisitos No Funcionales ("NFR") también se definen, ya que los requisitos que se expresan inicialmente como no funcionales conforme un proyecto progresa a menudo evolucionan parcial o totalmente en FUR que también puede ser medido.

Definir los principios básicos del método COSMIC, que se resumen en dos modelos.

'Modelo de Contexto de Software' se utiliza para caracterizar una pieza de software que se desea medir.

El 'Modelo Genérico Software' define los principios clave del modelo COSMIC de los FUR cuyo tamaño funcional se va a medir.

Para definir el proceso de medición del método COSMIC y el principio de medición (relacionado con la unidad de medición del método).

1.1 Aplicabilidad del Método COSMIC

El método COSMIC está diseñado para ser aplicable para medir la funcionalidad del software de los siguientes ámbitos:

Aplicaciones Software de Gestión, las cuales son las típicamente utilizadas para dar soporte a la administración de negocios, tales como banca, seguros, contabilidad, personal, compras, distribución o fabricación. Este software está a menudo caracterizado como "rico en datos", ya que está centrado principalmente en la necesidad de gestionar grandes cantidades de datos acerca de aspectos del mundo real.

Software en tiempo real, cuya tarea es mantener o controlar acontecimientos que están sucediendo en el mundo real. Algunos ejemplos serían el software para centrales telefónicas y de intercambio de mensajes, el software incluido en dispositivos para el control de máquinas tales como los electrodomésticos, ascensores, motores de automóviles y aeronaves, para el control de procesos y la adquisición automática de datos, y software de los sistemas operativos de los ordenadores.

El software de infraestructura en apoyo de lo anterior, tales como componentes reutilizables, controladores de dispositivos.

Algunos tipos de software científico / ingeniería.

1.2 Requisitos Funcionales de Usuario

El método de medición COSMIC consiste en la aplicación de un conjunto de modelos, principios, reglas y procesos a los Requisitos Funcionales de los Usuarios (o Functional User Requirements, FUR) de una determinada aplicación software. El resultado es un "valor de una cantidad" numérico (tal y como se define en ISO), que representa el tamaño funcional de la aplicación software de acuerdo con el método COSMIC.

Requisitos Funcionales de Usuario son definidos por la ISO [\[17\]](#) de la siguiente manera.

DEFINICIÓN – Requisitos Funcionales de Usuario (Functional User Requirements, FUR)

Subconjunto de los Requisitos de los Usuarios. Requisitos que describen lo que el software deberá hacer, en términos de tareas y servicios.

NOTA: Los requisitos funcionales de los usuarios incluyen, pero no están limitados a:

- Transferencia de datos (por ejemplo de entrada de datos de clientes; enviar señal de control)
- Transformación de datos (por ejemplo calcular los intereses bancarios; derivar temperatura media)
- Almacenamiento de datos (por ejemplo pedidos de clientes; recopilar datos de temperatura ambiente durante un tiempo)
- Recuperación de datos (por ejemplo listas de los empleados actuales; recuperar la última posición de una aeronave)

Ejemplos de requisitos de los usuarios que no son requisitos funcionales de los usuarios incluyen, pero no están limitados a:

- Restricciones de calidad (por ejemplo, facilidad de uso, fiabilidad, eficiencia y portabilidad)
- Restricciones de organización (por ejemplo, localización de operación, hardware objetivo y cumplimiento de normas)
- Restricciones medioambientales (por ejemplo, interoperabilidad, seguridad, privacidad y seguridad)
- Restricciones de Aplicación (por ejemplo, el lenguaje de desarrollo, calendario de entrega)

Tenga en cuenta que el método COSMIC reconoce que algunos tipos de requerimientos (por ejemplo, la calidad y las restricciones ambientales) pueden expresarse temprano en la vida de un proyecto de software como requerimientos 'No Funcionales', de acuerdo a la definición de la ISO. Sin embargo, estos mismos requisitos pueden evolucionar a medida que el proyecto avanza en Requerimientos Funcionales de Usuario. Vea la sección 1.2.3 de este Manual de Medición

NOTA: En este documento sólo utilizaremos el término 'FUR' para referirnos a los requerimientos funcionales de usuario que:

se derivan de los artefactos de software disponibles (requisitos, diseños, artefactos físicos, etc.)

se ajustan, de ser necesario, por supuestos para superar las incertidumbres en los artefactos disponibles,

contienen toda la información necesaria para una Medición de Tamaño Funcional con COSMIC. De lo contrario, vamos a utilizar 'necesidades reales' o 'artefactos físicos', etc., adecuadas al contexto.

El tamaño funcional medido por el método COSMIC está diseñado para depender sólo de los FUR del software que va a ser medido y ser independiente de cualquier requerimiento o restricciones relativas a la aplicación de los FUR. 'Funcionalidad' puede ser vagamente definido como 'el procesamiento de la información que el software debe realizar para sus usuarios'.

1.2.1 Extracción de los requisitos funcionales de usuario a partir de los artefactos del software en la práctica

En el mundo real de desarrollo de software es raro encontrar artefactos para el software en los que los FUR sean claramente distinguibles de otros tipos de requisitos y que sean expresados en un formato adecuado para realizar mediciones directas, sin necesidad de interpretación. Esto significa que por lo general el medidor tendrá que extraer los FUR tal y como se suministren, explícita o implícitamente, en

artefacto de software, antes de representarlos como conceptos de los “modelos de software” de COSMIC.

Como se ilustra en la figura 1. 1, los FUR pueden ser derivados de artefactos de ingeniería de software que se producen antes de que el software exista. Por lo tanto, el tamaño funcional del software puede medirse antes de su implementación como un sistema informático.

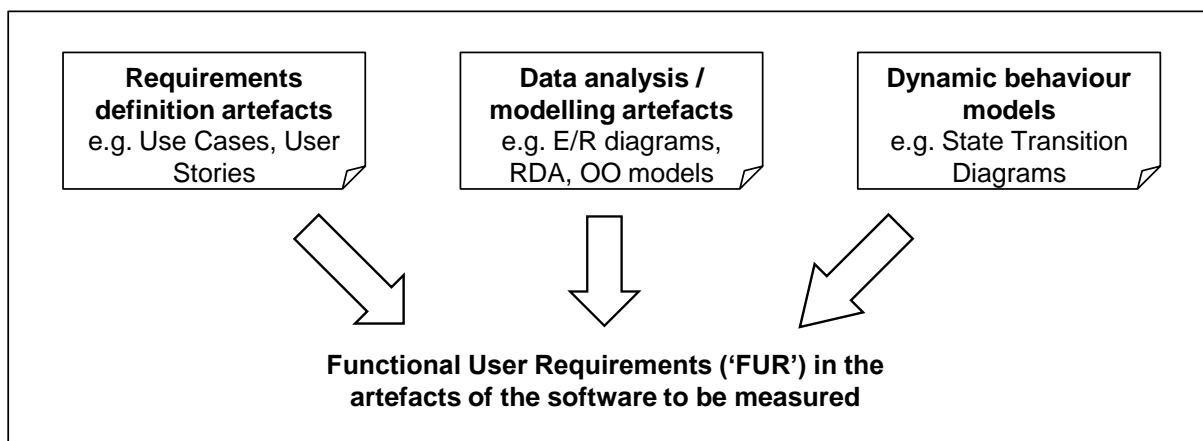


Figura 1. 1 – Fuentes de Pre-implementación de los Requisitos Funcionales de Usuario

NOTA: Los Requisitos Funcionales de Usuario se pueden producir incluso antes de que se coloquen en el hardware o software. Dado que el método COSMIC está dirigido a dimensionar el FUR de una pieza de software, sólo el FUR colocado en el software se mide. Sin embargo, en principio el método COSMIC puede ser aplicado a los FUR antes de que se coloquen en el software o hardware, independientemente de la decisión eventual de dónde se colocarán. Por ejemplo, es fácil determinar el tamaño de la funcionalidad de una calculadora de bolsillo utilizando el método COSMIC sin ningún conocimiento de qué el hardware o software (si existe) está involucrado. Sin embargo, la afirmación de que el método COSMIC se puede utilizar para determinar el tamaño los FUR colocado en el hardware necesita más pruebas en la práctica antes de que pueda ser considerado como totalmente validado y sin la necesidad de nuevas reglas.

En otras circunstancias, algún software existente puede necesitar ser medido sin que exista, o con sólo unos pocos artefactos disponibles de arquitectura o diseño, y los FUR pueden no estar documentados (por ejemplo, para el software de legado). En tales circunstancias, todavía es posible derivar los FUR a partir de los artefactos del sistema de cómputo, incluso después de que se ha implementado, como se ilustra en la Figura 1.2.

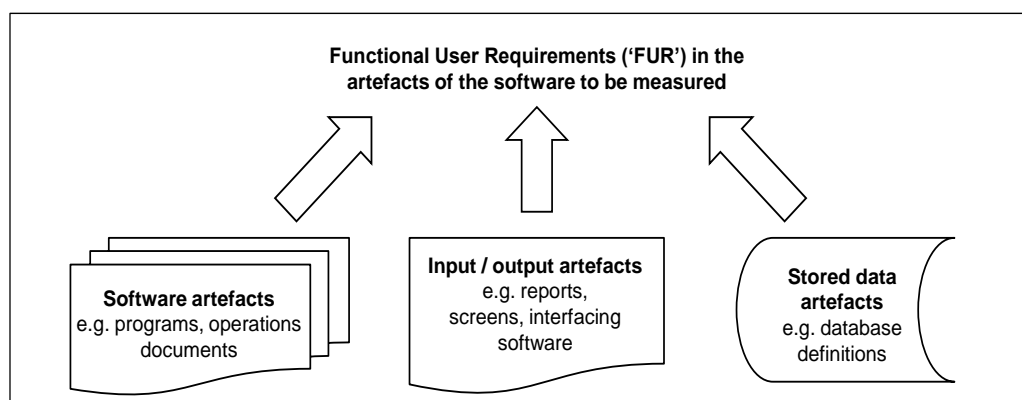


Figura 1.2 – Fuentes de Post-implementación de los Requisitos Funcionales de Usuario

1.2.2 Extracción o deducción de los requisitos funcionales de usuario a partir de artefactos software

Los procesos para ser utilizados y, por tanto, el esfuerzo necesario para extraer el FUR de diferentes tipos de artefactos de ingeniería de software o para derivar de ellos el software instalado y para su expresión en la forma requerida para la medición en el método COSMIC evidentemente variarán enormemente; estos procesos no pueden ser tratados en el Manual de Medición. Se asume que los requisitos funcionales del software que será medido bien existen o pueden ser extraídos o derivados de sus artefactos, a la luz del propósito de la medición

El Manual de Medición se limita a describir y definir los conceptos de los modelos COSMIC ('Modelo Contextual de Software' y 'Modelo Genérico de Software'- ver sección 1.3), y cómo aplicarlos para medir los FUR de una pieza de software.³

Si el medidor entiende realmente estos dos modelos, siempre será posible derivar los FUR de una pieza de software que se mide a partir de sus artefactos disponibles, aunque el medidor puede tener que hacer algunos supuestos debido a la poca clara o falta de información.

1.2.3 Requisitos NO-Funcionales (Non-Functional Requirements, NFR)

La definición de ISO para los Requisitos Funcionales de Usuario (o 'FUR', véase más arriba) enumera varios tipos de 'requisitos de usuarios' que no son FUR. Implícitamente se trata de Requisitos NO Funcionales (Non-Functional Requirements, NFR).

NFR puede ser muy significativo para un proyecto de software. En casos extremos, una especificación de los requisitos para un sistema de software intensivo puede requerir tanta documentación para los NFR como para los FUR. Pero la distinción entre NFR y de piel no es tan simple como se desprende de la definición de los FUR en ISO. El método COSMIC se puede utilizar para medir algunos de los requisitos que pueden expresarse primero como No funcionales. Primero tenemos que definir que son los NFR:

DEFINICION – Requisitos NO-Funcionales (Non-Functional Requirements) (de software)

Cualquier requisito para la parte de software de un sistema de software/hardware o producto de software, incluyendo la forma en que debe ser desarrollado y mantenido, y cómo debe desempeñarse en la operación, excepto algún requisito funcional de usuario para el software. Requisitos NO-Funcionales se refieren a:

- | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">• la calidad de software;• el medio ambiente en el que el software debe implementarse y que debe servir;• los procesos y la tecnología que se utilizará para desarrollar y mantener el software;• la tecnología que se utiliza para la ejecución de software |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

NOTA: Los requisitos del sistema o de software que se expresan inicialmente como No Funcional menudo evolucionan a medida que el proyecto avanza, total o parcialmente en FUR de un software.

Varios estudios [3] han demostrado que algunos de los requisitos que en un principio aparecen como requisitos NFR *de sistema* evolucionan a medida que el proyecto avanza en una mezcla de requisitos que pueden ser implementados en las funcionalidades del software y otros requisitos o restricciones que son verdaderamente 'no funcionales'. Ver Figura 1.3. Esto es cierto para muchas restricciones de calidad del sistema, tales como el tiempo de respuesta, facilidad de uso, mantenimiento, etc. Una vez identificadas, estas funciones de software que han sido 'ocultadas' en requisitos NFR en el comienzo de un proyecto se pueden dimensionar utilizando el método COSMIC como cualesquiera otras

³ Varias directrices COSMIC, p.e. para el dimensionamiento de software de aplicación empresarial [7] y para el dimensionamiento de software en tiempo real [4] dan orientación sobre el mapeo de varios análisis de datos y determinación de métodos de requisitos y de artefactos de software a los conceptos de COSMIC.

funcionalidades de software. No reconocer este tamaño funcional "oculto" es una razón por la cual el tamaño del software puede parecer que va creciendo como proyecto avanza.

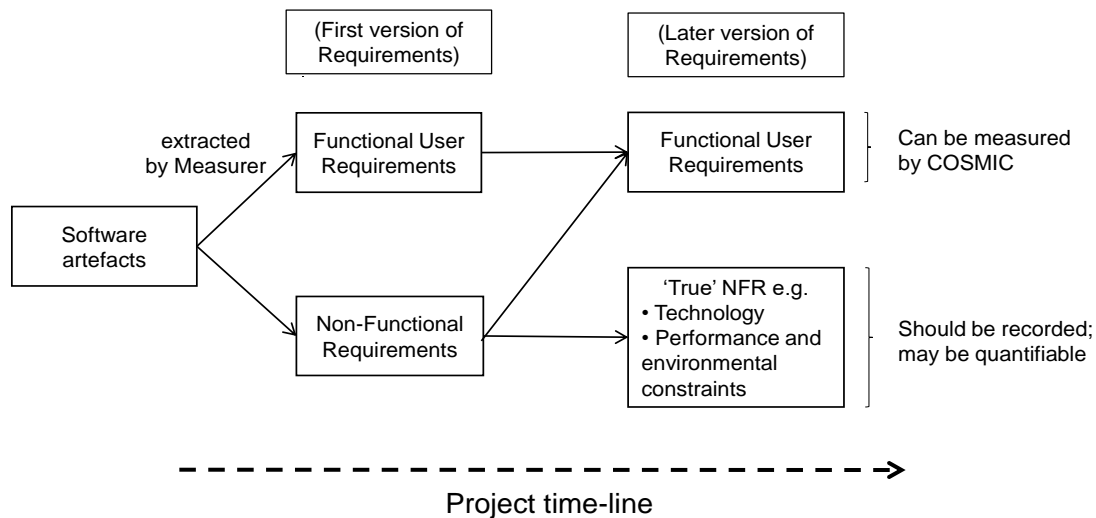


Figura 1.3 – Varios requisitos aparecen inicialmente como NFR evolucionan en FUR conforme avanza el proyecto

EJEMPLO APLICACIÓN DE NEGOCIO: Los requisitos para un nuevo sistema de software incluye la especificación "el usuario tendrá la opción de asegurar los archivos a través de encriptación". El proyecto para desarrollar el sistema está en la etapa de la estimación de esfuerzo y costo. Dos opciones se consideran:

Desarrollar un software de encriptación propietario. Para fines de estimación del proyecto puede ser necesario medir el tamaño de los FUR para el software de encriptado.

Compre un paquete existente Commercial Off-The-Shelf (COTS). Para fines de estimación del proyecto, puede ser necesario medir sólo el tamaño de la funcionalidad necesaria de software para integrar el paquete COTS que se va a adquirir. También tendrá que tener en cuenta en el cálculo del costo del proyecto el costo del paquete y el esfuerzo para integrar y probar el paquete de cifrado.

EJEMPLO EN TIEMPO REAL: La tolerancia a fallos en los sistemas aeroespaciales se logra principalmente a través de una combinación de redundancia y respaldo de los sistemas físicos. Una función como motor de monitoreo se lleva a cabo en dos o más computadoras embebidos separados. Esta función tiene una estricta restricción de tiempo indicado como NFR: 'cada equipo independiente debe responder en un tiempo específico. Si cualquiera de las computadoras responde repetidamente después del tiempo requerido, o sus resultados no están de acuerdo con los demás, debe ser eliminado (por un mecanismo especificado como un requisito funcional). Un requisito para la tolerancia a fallos cuando se especifica inicialmente puede aparecer como no funcional, se convierte en FUR que se puede medir. El mecanismo de tiempo también se puede implementar en parte en software y esta funcionalidad también se puede medir (véase por ejemplo la "Guía para el dimensionamiento de software en tiempo real" [4], sección 3.2).

Para más ejemplos, véase el apéndice B. Todos estos ejemplos demuestran que cuando hay un requisito para medir el tamaño de algún software en etapas tempranas del proyecto, es importante considerar si algún NFR podría convertirse en FUR y si el tamaño de éstos FUR también debe ser medido.

1.3 Los Principios Fundamentales del Método COSMIC

El método COSMIC se basa en sólidos principios de ingeniería de software. Estos principios se resumen en dos modelos.

De la misma manera que una casa puede tener muchos tamaños dependiendo de lo que se quiere medir, el tamaño de una pieza de software se puede medir de muchas maneras, incluso utilizando la misma unidad de medida. Los principios del 'Modelo Contextual de Software' permiten un medidor

definir el software a ser medido y lo que incluye la medición del tamaño. Con ello se garantiza que los resultados pueden ser entendidos e interpretados consistentemente por los futuros usuarios.

Los principios del 'Modelo Genérico de Software' definen cómo los FUR del software que se va a medir se modelan para que puedan ser medidos.

La razón principal para la inclusión de estos dos modelos en esta etapa en el Manual de Medición es mostrar cómo el método COSMIC es fundamentalmente muy simple. También se necesita para referirse a los dos modelos posteriormente en el Manual. Sin embargo, un medidor de novato no debe esperar ser capaz de leer estos dos modelos y luego desaparecer y medir con precisión. Para aplicar los modelos a una situación particular de medición, el medidor tendrá las definiciones de los diversos conceptos y los principios adicionales, reglas, explicaciones y ejemplos que se dan en este manual.

N.B. Los términos que se dan en **negrita** cuando se utilizan por primera vez en las siguientes secciones 1.3.1 y 1.3.2 son usados con significados que pueden ser específicos al método COSMIC. Para las definiciones formales, consulte el glosario al final de este Manual de Medición. Las referencias dadas con cada principio son las secciones donde el tema es tratado en detalle dentro de este manual.

1.3.1 El Modelo Contextual de Software de COSMIC

PRINCIPIOS – Modelo Contextual de Software de COSMIC	Sección
a) El software está limitado por el hardware.	-
b) El software está típicamente estructurado en capas .	2.2.2
c) Una capa puede contener una o más aplicaciones de software semejante	2.2.2
d) Cualquier aplicación software que deba medirse, se define por su alcance de medición, que se limitará en su totalidad dentro de una sola capa.	2.2
e) El alcance de la aplicación de software debe medirse en función del propósito de la medición.	2.1
f) Los usuarios funcionales de una aplicación de software se identificarán a partir de los FUR de la aplicación software que se medirá como los emisores y/o destinatarios de los datos al/desde el software respectivamente.	2.3
g) Los FUR de software pueden expresarse en distintos niveles de granularidad .	2.4
h) Una medición precisa en COSMIC del tamaño de una pieza de software requiere que sus FUR sean conocidos en un nivel de granularidad en el que se puedan identificar sus procesos funcionales y subprocesos.	2.4.3
i) Una medición aproximada en COSMIC de una pieza de software es posible si sus FUR se miden a un alto nivel de granularidad por un enfoque de aproximación y es escalado al nivel de granularidad de procesos funcionales y subprocesos.	2.4.3

1.3.2 El Modelo Genérico de Software

Después de haber identificado y definido los FUR del software a ser medido en términos del “Modelo Contextual del Software”, debemos ahora aplicar el Modelo Genérico del Software a los FUR para identificar los componentes de la funcionalidad que se medirán. Este Modelo Genérico de Software asume que los siguientes principios generales son ciertos para cualquier software que puede ser medido con el método.

PRINCIPIOS – Modelo Genérico de Software de COSMIC	Sección
a) Una pieza de software interactúa con sus usuarios funcionales a través de una frontera , y con un almacenamiento persistente que existe dentro de esa frontera.	2.3
b) Los requisitos funcionales de los usuarios de la aplicación software a medir pueden representarse como procesos funcionales únicos.	3.2
c) Cada proceso funcional se conforma por sub-procesos.	3.2
d) Un sub-proceso puede ser un movimiento de datos o una manipulación de datos .	3.2
e) Un movimiento de datos mueve un sólo grupo de datos .	3.3
f) Existen cuatro tipos de movimientos de datos: <ul style="list-style-type: none"> • Una Entrada, mueve un grupo de datos hacia un proceso funcional desde un usuario funcional. • Una Salida mueve un grupo de datos fuera de un proceso funcional hacia un usuario funcional. • Una Escritura mueve un grupo de datos desde un proceso funcional hacia un almacén persistente. • Una Lectura mueve un grupo de datos desde un almacén persistente hacia un proceso funcional 	3.5
g) Un grupo de datos consiste en un conjunto único de atributos de datos que describen un único objeto de interés .	3.4
h) Cada proceso funcional es activado por un movimiento desencadenante de entrada de datos. El grupo de datos movido por el evento desencadenante es generado por un usuario funcional en respuesta a un evento desencadenante .	3.2
i) Un proceso funcional deberá incluir al menos un movimiento de entrada y también un movimiento de Salida o Escritura de datos, es decir que deberá incluir un mínimo de dos movimientos de datos. No existe un límite superior para el número de movimientos de datos en un proceso funcional.	3.5
j) Como una aproximación para fines de medición, los sub-procesos de manipulación de datos no se miden por separado; la funcionalidad de cualquier manipulación de datos se supone que se tomada en cuenta por el movimiento de datos con el que está asociado.	

1.3.3 Tipos contra Ocurrencias

Para entender este Manual de Medición, es esencial que el lector puede distinguir entre 'tipos' y 'ocurrencias'.

Todos los métodos de medición tamaño funcional definen los 'tipos de cosas' que un medidor debe identificar en los requisitos funcionales dados con el fin de medir un tamaño funcional. En el caso del método COSMIC, estos 'tipos de cosas' incluyen tipos funcionales de los usuarios y tipos de procesos funcionales desde el Modelo Contextual de Software y todos los conceptos que aparecen en negritas, por ejemplo, los tipos de datos de movimiento, tipos de objeto de interés, etc., en el Modelo Genérico de Software.

Para facilitar la lectura, sin embargo, normalmente omitimos "tipo" cuando se utilizan estos términos.

En general:

el 'tipo' de una cosa es una clase abstracta de todas las cosas que comparten alguna característica común. (Sinónimos de 'tipo' son 'categoría' o 'clase')

una 'ocurrencia' de una cosa es cuando la cosa aparece en la práctica, por ejemplo, en un contexto del mundo real, o cuando se produce un evento, o cuando un proceso se ejecuta por una persona o una computadora. (Un sinónimo de 'ocurrencia' es 'instancia'). Se crea una 'ocurrencia', cuando las características de un 'tipo' de algo se dan valores reales, conocidos en el mundo orientado a objetos como 'instancias' -. La creación de una instancia)

Ejemplos del Modelo Contextual de Software

EJEMPLO 1: El sistema de apoyo a un Centro de Atención Telefónica tiene 100 empleados que contestan preguntas de los clientes. Un modelo contextual del sistema debería mostrar un tipo de usuario funcional: 'empleado' de los cuales hay 100 ocurrencias

EJEMPLO 2: El software embebido de un radio digital envía su salida a un par de altavoces estereofónicas funcionalmente idénticas. Un modelo contextual de software debería mostrar un usuario funcional del tipo de 'altavoz' de los cuales hay dos ocurrencias.

Ejemplos del Modelo Genérico de Software

EJEMPLO 3: Supongamos que un proceso funcional (- tipo) que permite que los datos sean introducidos y validados para un nuevo cliente. El proceso funcional será ejecutado, es decir, que ocurrirá, cada vez que un usuario funcional humano registre los datos para un nuevo cliente específico. Sin embargo, durante su ejecución, el movimiento de datos de Lectura del proceso que debe validar los datos introducidos mediante una búsqueda para comprobar si el cliente ya existe en la base de datos puede ocurrir una o más veces (dependiendo del diseño de base de datos).

EJEMPLO 4: Supongamos que un proceso funcional (- tipo) debe controlar la temperatura de un horno una vez cada diez segundos. El proceso funcional será ejecutado, es decir, ocurrirá, una vez cada 10 segundos. Durante su ejecución, el movimiento de datos de Salida del proceso funcional para cambiar el calentador encendido o apagado puede o no ocurrir en cualquier ciclo, dependiendo de si el calentador debe estar encendido o apagado, o se deja en su estado actual.

Nota: El número de ocurrencias de cualquier usuario funcional, o de cualquier movimiento de datos de Entrada, Salida, Lectura o Escritura cuando el software se ejecuta es totalmente irrelevante para la medición de un tamaño funcional COSMIC.

1.4 El Proceso de Medición COSMIC y la Unidad de Medida

El proceso de medición COSMIC se compone de tres fases:

La Fase de Estrategia de Medición, en la cual el propósito y alcance de la medición es definido. El Modelo Contextual de Software se aplica entonces para que el software que se va a medir y la medición requerida sea definida sin ambigüedad. (Capítulo 2)

La Fase de Mapeo (Representación) de la Medición, en la cual el Modelo Genérico de Software se aplica a los FUR del software que se va a ser medido para producir el modelo COSMIC del software que pueda ser medido. (Capítulo 3)

La Fase de Medición, en la que se miden los tamaños reales. (Capítulo 4) Las Reglas de cómo deben registrarse las mediciones se dan en el Capítulo 5.

La relación de las tres fases del método COSMIC se muestra en la Figura 1.4:

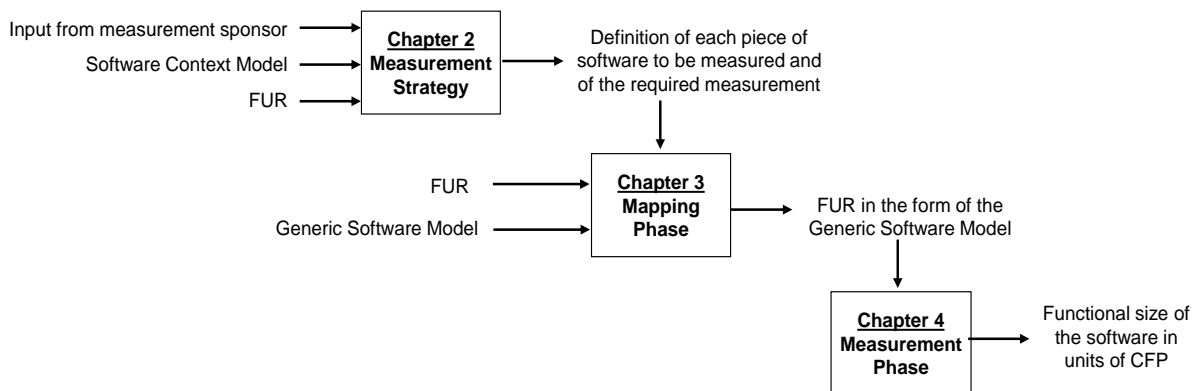


Figura 1.4 – El Proceso de Medición del Método COSMIC

La unidad de medida COSMIC (el ‘CFP’) y el principio de medición se definen de la siguiente manera.

DEFINICION – Unidad de Medida COSMIC
1 CFP (Punto Funcional Cosmic, <u>C</u> osmic <u>F</u> unction <u>P</u> oint), que es el tamaño de un movimiento de datos.

PRINCIPIO – El Principio de Medición COSMIC
El tamaño de un proceso funcional es igual al número de sus movimientos de datos. El tamaño funcional de una pieza de software de alcance definido es igual a la suma de los tamaños de sus procesos funcionales.

El tamaño de los cambios necesarios en una pieza de software se mide de la siguiente manera:

- El tamaño de cualquier movimiento de datos afectado (es decir, que hay que añadir, modificar o borrar) por el cambio requerido se mide por convención como uno CFP.
- El tamaño de los cambios requeridos en una pieza de software es igual al número de movimientos de datos que se ve afectada por los cambios requeridos.
- El tamaño mínimo de un cambio en una pieza de software es 1 CFP.

Otras reglas y guías sobre la medición y la adición de mediciones se explican en las secciones 4.1 a 4.4 del presente Manual de Medición.

1.5 Limitaciones sobre la Aplicabilidad del Método COSMIC

Véase la sección 4.5 para posibles limitaciones del método y cómo puede ser posible extender el método localmente para superar las limitaciones.

LA FASE DE ESTRATEGIA DE MEDICIÓN

2.0 Resumen del Capítulo

En este capítulo se describen los parámetros clave que deben ser considerados en la primera fase del proceso de medición denominada 'Estrategia de Medición', antes de realmente comenzar a medir. Estos son (*en itálicas*):

- El *propósito* de la medición, es decir, para lo que será utilizado el resultado. El propósito determina los otros parámetros de una medición.
- El *alcance global* del software a ser medido y, si el software se compone de más de una parte que deba ser medida por separado (por ejemplo, los componentes de un sistema de software distribuido), los *alcances de las mediciones* de las partes individuales. También tenemos que determinar la *capa* en la que cada pieza de software se encuentra y tal vez el *nivel de descomposición* de las piezas de software que se desean medir.
- Los *usuarios funcionales* de cada pieza de software a medir. Estos son los emisores y destinatarios de los datos a/desde el software que se desea medir; pueden ser seres humanos, dispositivos de hardware u otras piezas de software. Como los diferentes usuarios funcionales pueden tener requisitos para los diferentes subgrupos de la misma funcionalidad global, el tamaño funcional variará con la elección de los usuarios funcionales.
- El *nivel de granularidad* de los artefactos disponibles del software que se va a medir. Por ejemplo, tal vez la única especificación de los requisitos no se define en todo el detalle necesario para una medición precisa con el método COSMIC. Por tanto, debemos decidir cómo derivar los FUR a medir y/o si una variante de aproximación de tamaño se debería utilizar.

La determinación de estos parámetros ayuda a responder a las preguntas de 'qué tamaño debe medirse', 'qué tan preciso queremos la medición', etc. El registro de los parámetros habilita a los futuros usuarios de una medición para decidir la forma de interpretarla.

Es importante señalar que estos parámetros y los conceptos relacionados no son específicos para el método de Medición de Tamaño Funcional (FSM) COSMIC, pero deberían ser comunes a todos los métodos de FSM. Otros métodos de FSM no pueden distinguir diferentes tipos de usuarios funcionales y no pueden discutir los diferentes niveles de granularidad, etc. La aplicación más amplia y la flexibilidad del método COSMIC es la que requiere que estos parámetros sean considerados con más cuidado que con otros métodos FSM.

Es muy importante registrar los datos que surgen de esta Fase Estrategia de Medición (como se indica en la sección 5.2) cuando se registra el resultado de cualquier medición. Fallas al definir y registrar estos parámetros consistentemente dará lugar a mediciones que no puedan ser interpretados de forma confiable y compararse, o utilizarse de forma confiable como insumo para procesos tales como la estimación de esfuerzo del proyecto.

Las secciones de este capítulo proporcionan las definiciones formales, principios, reglas y ejemplos para cada uno de los parámetros clave para ayudar al medidor a través del proceso de determinación de una estrategia de medición, como se muestra en la Figura 2.0 a continuación.

Cada sección ofrece una explicación de antecedentes de por qué el parámetro clave es importante, usando analogías para demostrar por qué se toma el parámetro como garantía en otros campos de la medición, y por lo tanto también debe ser considerado en el ámbito de la medición de tamaño funcional del software.

Nótese de la figura 2.0 que la determinación de los parámetros de la estrategia de medición puede necesitar alguna iteración

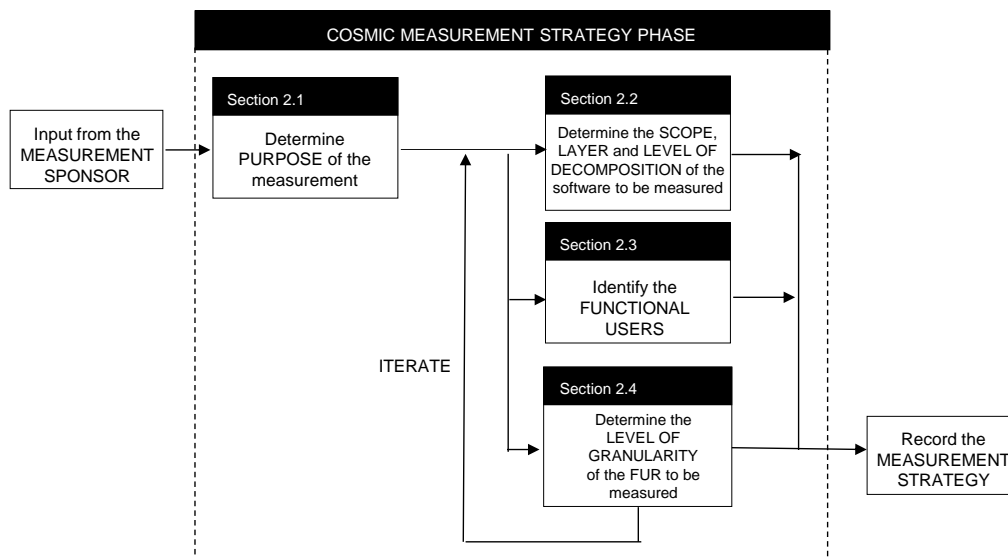


Figura 2.0 – El Proceso para determinar una Estrategia de Medición

Patrones de Estrategia de Medición

Como ayuda para determinar una estrategia de medición, la Guía para 'Patrones de Estrategia de Medición' [5] describe, para varios tipos diferentes de software, un conjunto estándar de parámetros para medir el tamaño de software, llamado 'Patrón de Estrategia de Medición' (abreviado a 'Patrón de Medición').

DEFINICIÓN – Patrón de Medición (Estrategia)

Una plantilla estándar que se puede aplicar en la medición de una pieza de software desde un ámbito funcional de software determinado, que define los tipos de usuario funcional que pueden interactuar con el software, el nivel de descomposición del software y los tipos de movimientos de datos que el software puede manejar.

El uso constante de los mismos patrones de medida debería ayudar Medidores para asegurar que las mediciones realizadas con el mismo propósito se realizan de una manera consistente, se puede comparar de forma segura con otras mediciones hechas usando el mismo patrón y se interpretarán correctamente para todos los usos futuros. Un beneficio adicional de la utilización de un patrón estándar es que el esfuerzo para determinar los parámetros de la estrategia de medición se reduce mucho. Nosotros, sin embargo, recomendamos que los Medidores estudien y dominen el método COSMIC, especialmente los parámetros de la Estrategia de Medición, antes de utilizar los patrones estándar.

2.1 Definir el propósito de la Medición

El término 'propósito' es usado en su significado inglés.

DEFINICIÓN – Propósito de la medición

Una declaración que define por qué una medición es necesaria, y para qué se utilizará el resultado

2.1.1 *El propósito de una medición determina el tamaño que será medido*

Hay muchas razones para medir el tamaño funcional del software, así como hay muchas razones para medir, por ejemplo, las áreas superficiales de una casa y en ambos casos las diferentes razones pueden resultar en diferentes tamaños. A partir de esta analogía, el tamaño de una casa bien puede variar con:

- la razón y el momento de la medición (por ejemplo, en función de la necesidad de medir las especificaciones del cliente para estimar el presupuesto, o los planos del arquitecto para una estimación precisa de costos, o el tamaño real para la planificación en la terminación de revestimientos de suelo),
- los artefactos de medición (por ejemplo, los planos o el edificio físico)

Tenga en cuenta, sin embargo, que los mismos principios de medición y la unidad de medida se utilizan para todas las mediciones. Exactamente de la misma manera, el tamaño medido de una pieza de software puede variar con:

- la razón y el momento de la medición (por ejemplo, en función de la necesidad de medir antes del desarrollo para fines de estimación, o durante el desarrollo para controlar el alcance ('scope creep'), o después de la instalación para medir el desempeño de los desarrolladores),
- los artefactos medidos (por ejemplo, una especificación de requisitos, o los artefactos físicos de software).

Como en la analogía, sin embargo, los mismos principios de medición y la unidad de medida se utilizan para todas las mediciones.

Es evidente que el Medidor de una pieza de software debe decidir, en función de la finalidad de la medición, *cuándo* medir (antes, durante o después del desarrollo), *qué* medir (por ejemplo, todo el software que se entrega en un proyecto, o excluir el software reutilizado) y *cuáles artefactos* utilizar para derivar los FUR a ser medidos (por ejemplo, una especificación de requisitos o el software instalado).

EJEMPLOS: Los siguientes son los propósitos de medición típicos

- *Para medir el tamaño de los FUR a medida que evolucionan, como entrada a un proceso para estimar el esfuerzo de desarrollo.*
- *Para medir el tamaño de los cambios a los FUR después de que se hayan establecido inicialmente, con el fin de gestionar el alcance 'scope creep'.*
- *Para medir el tamaño de los FUR del software entregado como entrada para la medición del desempeño de la organización de desarrollo.*
- *Para medir el tamaño de los FUR del software entregado, y también el tamaño de los FUR del software que fue desarrollado, con el fin de obtener una medida de la reutilización funcional.*
- *Para medir el tamaño de los FUR del software existente como entrada para la medición del desempeño del grupo responsable de mantener y dar soporte al software.*
- *Para medir el tamaño de algunos cambios a (los FUR de) un sistema de software existente como una medida del tamaño del trabajo a realizar de un equipo de proyecto de mejora.*
- *Para medir el tamaño del subconjunto de la funcionalidad total del software que debe ser desarrollado, y que se proporcionará a los usuarios funcionales seres humanos del software.*

2.1.2 *La importancia del propósito*

El propósito ayuda a los medidores a determinar:

- El alcance que debe medirse y por lo tanto los aparatos que serán necesarios para la medición.
- Los usuarios funcionales (como se indica en la sección 2.3, el tamaño funcional de los cambios depende sobre quién o qué es definido como usuario funcional).
- El momento en el ciclo de vida del proyecto en el que la medición se llevará a cabo
- La precisión necesaria de la medición, y por lo tanto, si debería utilizarse la medición con el método COSMIC estándar, o si se debería utilizar una aproximación del método COSMIC (por ejemplo, en

etapas tempranas del ciclo de vida de un proyecto, antes de que los FUR estén totalmente elaborados).

Los dos últimos puntos determinarán el nivel de granularidad al cual serán medidos los FUR.

2.2 Definir el alcance de la medición

DEFINICIÓN – Alcance de una medición

El conjunto de los FUR que deben incluirse en un determinado ejercicio de medición de tamaño funcional.

NOTA: (específico para el método COSMIC) Se debe hacer una distinción entre el 'alcance global', es decir, todo el software que debe medirse de acuerdo al propósito, y el 'alcance' de cualquier pieza de software individual dentro del alcance global, cuyo tamaño debe medirse por separado. En este Manual de Medición, el término 'alcance' (o la expresión 'alcance de la medición') se referirán a una pieza individual de software cuyo tamaño debe medirse por separado.

REGLAS – Alcance de la Medición

- a) El alcance de cualquier pieza de software a ser medida se deriva del propósito de la medición.
- b) El alcance de cualquier medición no se extenderá por más de una capa de software que se desea medir.

Consulte la siguiente sección para obtener ejemplos del alcance global y alcances de medición.

2.2.1 Derivando el alcance a partir del propósito de una medición

El software definido por un *alcance global* puede ser sub-dividida en piezas individuales de software cada uno con su propio *alcance de medición* definido de muchas maneras, dependiendo del propósito de la medición. Supongamos que un alcance global se define como 'el portafolio de aplicaciones de la organización X' o como 'todas las piezas de software que se entregarán por el proyecto Y'. Las subdivisiones se podrían hacer debido a:

- El software en diferentes capas (debido a la regla b) anterior),
- diferentes responsabilidades de la organización, por ejemplo, por grupo de clientes o por equipo asociado a sub-proyecto,
- la necesidad de distinguir los diferentes entregables para la medición del desempeño, la estimación de esfuerzo o con fines contractuales del software.

La última razón podría ser debido a la necesidad de distinguir alcances de medición separados para piezas de software que:

- se construyen utilizando diferentes tecnologías, es decir, la plataforma de hardware, lenguaje de programación, etc.,
- operar en diferentes modos, es decir, en línea versus modos batch,
- se desarrolló como contraposición al 'entregado' (incluido el último. Implementado un paquete u otro software reutilizado),
- se encuentran en diferentes niveles de descomposición, por ejemplo, una aplicación total o un componente principal o un componente menor, como objeto reutilizable,

- son los principales entregables en contraposición al software que se utiliza una vez, por ejemplo, para la conversión de datos y, a continuación son descartados; tal vez no valga la pena el esfuerzo de realizar la medición después.
 - se entregan por 'sprints' individuales de un proceso ágil
 - son desarrollados en lugar de mejorar el software,
- y cualquier combinación de estos factores.

En resumen, el propósito de la medición debe ser siempre utilizado para determinar (a) qué software es incluido o excluido del alcance global y (b) la forma en que el software incluido puede que sea necesario dividirlo en partes separadas, cada una con su propio alcance, que deben medirse por separado.

En la práctica, una especificación de alcance debe ser explícita en lugar de genérica, por ejemplo, el trabajo-producto desarrollado del equipo del proyecto 'A', o la aplicación 'B', o el portafolio de la empresa 'C'. La especificación de alcance puede también, establecer lo que se excluye, para mayor claridad.

EJEMPLO APLICACIÓN DE NEGOCIO: Figura 2.1 muestra todas las piezas de software separadas - el 'alcance global' - entregado por un equipo de proyecto:

- el cliente y los componentes de servidor de un paquete de software implementado
- un programa que proporciona una interfaz entre el componente de servidor del nuevo paquete y las aplicaciones existentes
- un programa que se utiliza una vez para convertir los datos existentes al nuevo formato requerido por el paquete. Este programa ha sido desarrollado usando un número de objetos reutilizables desarrollados por el equipo del proyecto
- el software del controlador de dispositivo para el nuevo hardware en el que el componente del paquete del cliente se ejecutará

(Cada pieza individual de software para ella cual se define un alcance de la medición se muestra como una caja rectangular sólida.)

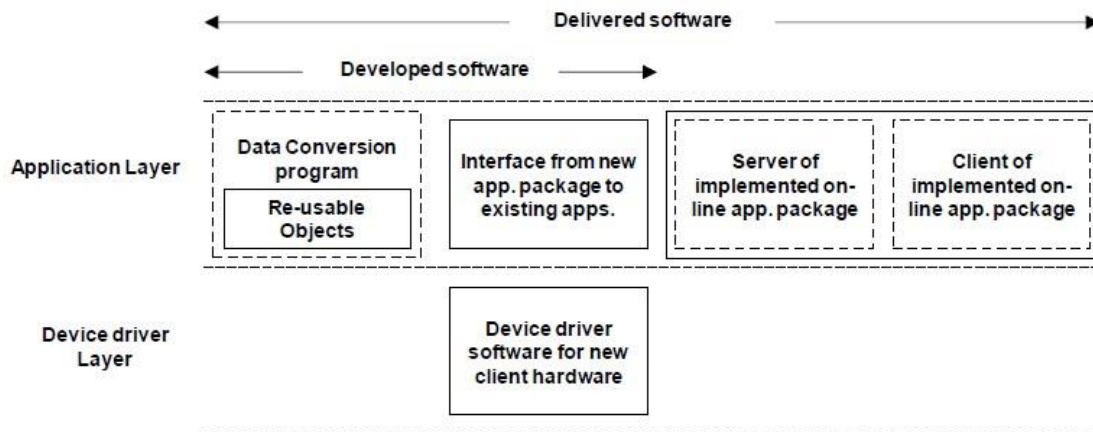


Figura 2.1 - The overall scope of the deliverables of a software project and the individual measurement scopes

El diagrama muestra que las piezas de software 'entregadas' consistieron en algunas que fueron desarrollos nuevos y algunas que fueron implementadas por el equipo del proyecto.

El tamaño del paquete implementado se midió como un 'todo', es decir, haciendo caso omiso de la estructura de los componentes cliente-servidor. Este tamaño se agregó al programa de interfaz para actualizar el tamaño total del portafolio de aplicaciones de la organización. El tamaño del programa de conversión de datos no era de interés, ya que solo se usa una vez y se tira. Pero el tamaño de cada uno de los objetos reutilizables se registró en el inventario de software de infraestructura de la organización, así como la del nuevo controlador de dispositivo. Una vez más estos se clasificaron por separado.

Debido a la naturaleza diversa de los entregables no sería sensato al medir el desempeño del equipo del proyecto, sumar los tamaños de todo el software entregado. El rendimiento de los equipos que entregan cada pieza de software se debe medir por separado.

2.2.2 Capas

Dado que el alcance de una pieza de software que debe medirse debe limitarse a una sola capa de software, el proceso para definir el alcance podrá requerir que el medidor primero tenga que decidir cuáles son las capas de la arquitectura del software. Por lo tanto en esta sección vamos a definir y discutir “capas” de software y como estos términos se utilizan en el método COSMIC. Las razones por las que necesitamos estas definiciones y reglas son las siguientes:

- El medidor puede encontrarse con la medición de algún tipo de software en un entorno de software “legado” que ha evolucionado a lo largo de muchos años sin haber sido diseñado de acuerdo a una arquitectura subyacente (la llamada arquitectura espagueti) El medidor puede, por tanto, necesitar orientación de cómo distinguir las capas de acuerdo con la terminología COSMIC.
- Las expresiones ‘capa’ y ‘arquitectura en capas’ no se utilizan consistentemente en la industria del software. Si el medidor debe medir algún tipo de software que ha sido descrito mediante una “arquitectura de capas”, es aconsejable comprobar que las “capas” en esta arquitectura están definidas de una manera que sea compatible con el método COSMIC. Para ello, el medidor debería establecer la equivalencia entre objetos arquitectónicos específicos en la ‘arquitectura de capas’ y el concepto de capas, tal como se definen en este manual.

Las capas pueden ser identificadas conforme a las siguientes definiciones y principios

DEFINICIÓN – Capa
Una partición funcional de una arquitectura de un sistema de software.

En una arquitectura de software definida, cada capa debe cumplir con los siguientes principios:

PRINCIPIOS – Capa
<p>a) El software en una capa proporciona un conjunto de servicios que es coherente de acuerdo con algún criterio definido, y ese software puede utilizarse en otras capas sin saber cómo se implementan estos servicios.</p> <p>b) La relación entre el software en cualquiera de dos capas se define por una 'regla de correspondencia' que puede ser</p> <ul style="list-style-type: none"> • ‘jerárquica’, es decir, el software en la capa A tiene permitido utilizar los servicios proporcionados por el software en la capa B, pero no al revés (donde la relación jerárquica puede ser hacia arriba o hacia abajo), o • 'bidireccional', es decir, el software en la capa A tiene permitido utilizar el software en la capa B, y viceversa. <p>d) El Software en una capa intercambia grupos de datos con el software en otra capa a través de sus respectivos procesos funcionales.</p> <p>e) El Software en una capa no necesariamente utiliza todos los servicios funcionales proporcionados un software en otra capa.</p> <p>f) El Software en una capa de una arquitectura de software definida puede ser dividido en otras capas de acuerdo a diferentes arquitecturas de software definidas.</p>

Una medición puede referirse a dos o más piezas 'semejantes' de software, que se definen de la siguiente manera:

DEFINICIÓN – Componentes Semejantes de Software o Pares (Peer)

Dos piezas de software son semejantes una a la otra si se encuentran en la misma capa.

EJEMPLO: Las piezas de software en la capa de aplicación de la Figura 2.1 son todos semejantes entre sí.

Si el software a ser medido existe dentro de una arquitectura de capas establecida que se puede mapear a los principios de capas de COSMIC como los definidos anteriormente, entonces, la arquitectura debe ser usada para identificar las capas para fines de medición.

Si el propósito requiere que algún software que se mide y que no está estructurado de acuerdo con los principios de capas de COSMIC, el medidor debe tratar de dividir el software en capas mediante la aplicación de los principios definidos anteriormente. Convencionalmente, paquetes de infraestructura de software como los sistemas de gestión de bases de datos, sistemas operativos o controladores de dispositivos, que proporcionan servicios que pueden ser utilizados por otros programas en otras capas, se considera que cada uno se encuentran en capas separadas.

Normalmente, en las arquitecturas de software, la capa más 'superior', es decir, la capa que no es un subordinado a ninguna otra capa en una jerarquía de capas, que se conoce como la capa de 'aplicación'. El software en esta capa de aplicación se basa en los servicios de software de todas las otras capas para que se ejecute desempeñe correctamente. Software en esta capa 'superior' puede ser en sí mismo dividido en capas, por ejemplo, como en una 'arquitectura de tres capas' de la interfaz de usuario, reglas de negocio y los componentes de servicios de datos (véase el Ejemplo 5 más adelante).

Una vez identificadas, cada capa puede ser registrada en la matriz del Modelo Genérico de Software (Apéndice A), con la etiqueta correspondiente.

APLICACIÓN DE NEGOCIO EJEMPLO 1: La estructura física de una típica arquitectura de software en capas (utilizando el término 'capa' como ha sido definido aquí) que soporte software de aplicaciones de negocio se da en la figura 2.2:

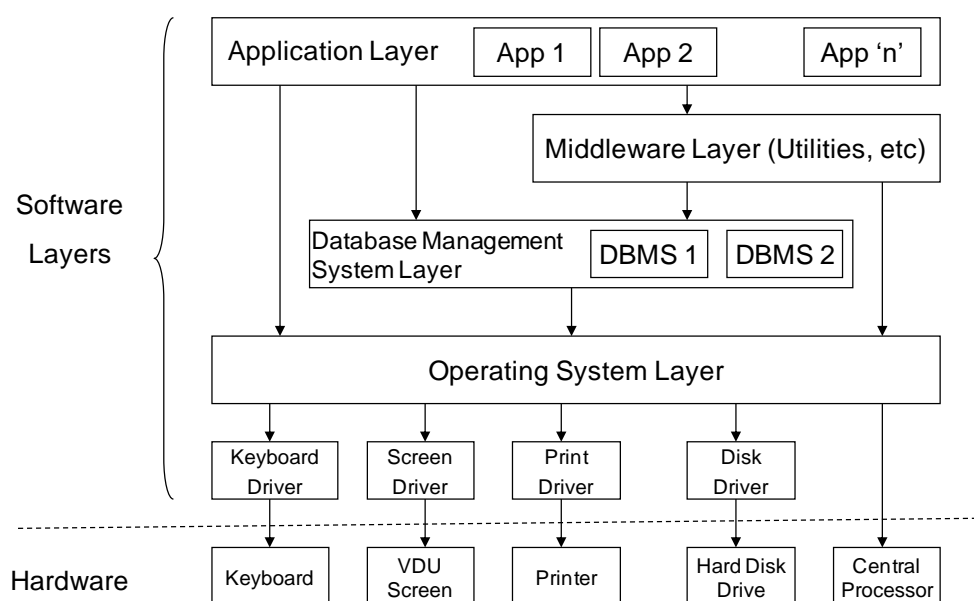


Figura 2.2- Típica arquitectura de software en capas de una aplicación de negocios

TIEMPO REAL EJEMPLO 2: La estructura física de una típica arquitectura de software en capas (de nuevo utilizando el término ‘capa’ como ha sido definido aquí) soportando una pieza de software integrada (embebida) de tiempo real se muestra en la figura 2.3:

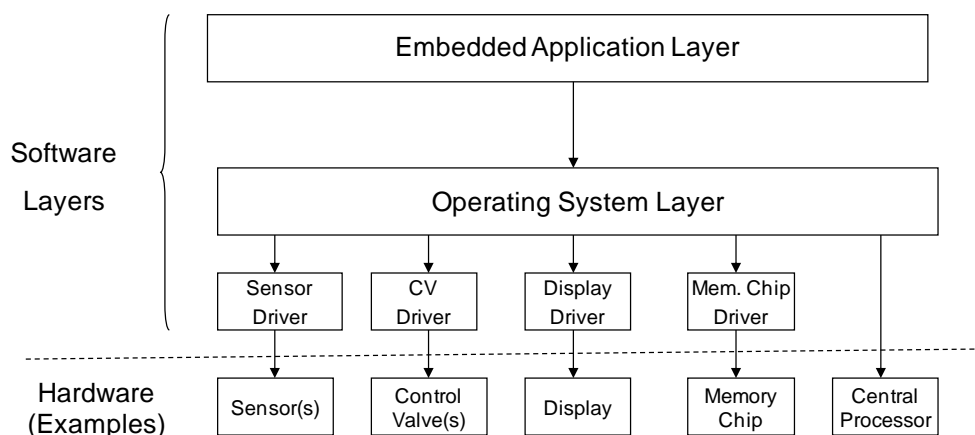


Figura 2.3 - Típica arquitectura en capas de un sistema software embebido de tiempo real

TIEMPO REAL EJEMPLO 3: El modelo ISO de 7 capas (OSI) para las telecomunicaciones. Esto define una arquitectura en capas para que las reglas jerárquicas correspondientes para las capas del software de recepción de mensajes son inversas a las reglas para las capas del software de transmisión de mensajes.

TIEMPO REAL EJEMPLO 4: La arquitectura 'AUTOSAR' de la industria automotriz que exhibe todos los diferentes tipos de reglas de correspondencia entre las capas ahora descritos en los principios de una capa.

Ver: [http://www.autosar.org/fileadmin/files/releases/4-](http://www.autosar.org/fileadmin/files/releases/4-2/softwarearchitecture/general/auxiliary/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf)

[2/softwarearchitecture/general/auxiliary/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf](http://www.autosar.org/fileadmin/files/releases/4-2/softwarearchitecture/general/auxiliary/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf).

Una arquitectura de software puede presentar diferentes capas en función de la ‘vista’ de la arquitectura.

APLICACIÓN DE NEGOCIO EJEMPLO 5: Considere una aplicación A situada en una arquitectura de software en capas, como se muestra en la figura 2.4 abajo, que muestra tres posibles estructuras de capas a), b) y c) de acuerdo con diferentes ‘vistas’ de arquitectura.

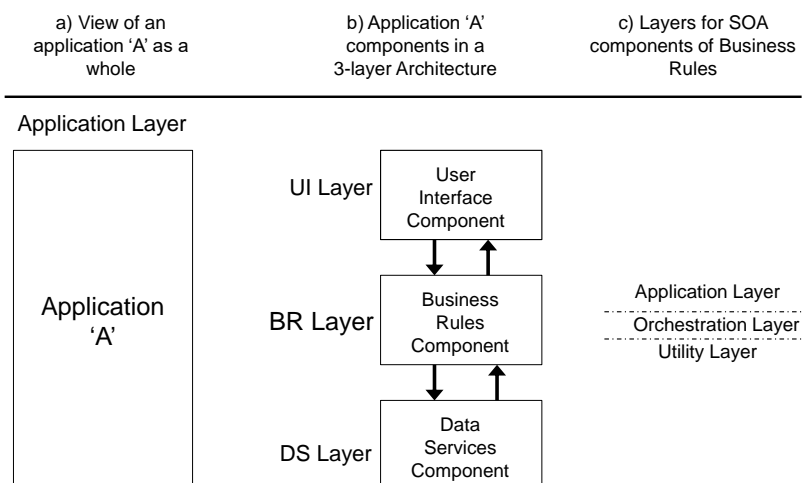


Figura 2.4 - Tres vistas de capas de una aplicación

Propósito 1 es medir el tamaño funcional de la aplicación A 'en su conjunto', como en la Vista a) Figura 2.4. El alcance de la medición es la aplicación A en su conjunto, que existe completamente dentro de una capa de 'aplicación'

Propósito 2. La aplicación A se ha construido de acuerdo con una arquitectura 'de tres capas' que comprende una interfaz de usuario, reglas de negocio y los componentes de servicios de datos. El propósito 2 es medir los tres componentes por separado como en la Vista b) Figura 2.4. Cada componente se encuentra en su propia capa de la arquitectura y el alcance de la medición se debe definir por separado para cada componente.

Propósito 3. El componente Reglas de Negocio de la aplicación se ha construido utilizando componentes reutilizables de una Arquitectura Orientada a Servicios (SOA), que tiene su propia estructura de capas. El propósito 3 es medir un componente SOA del componente de Reglas de Negocio como en la Vista c) Figura 2.4. Cada componente SOA está situado en una capa de la arquitectura SOA y el alcance de la medición se debe definir por separado para cada componente SOA. (Tenga en cuenta que la terminología SOA también utiliza 'capa de aplicación' dentro de su propia arquitectura.)

2.2.3 Niveles de descomposición

El 'nivel de descomposición' de una pieza de software se define como sigue:

DEFINICIÓN – Nivel de descomposición
<p>Cualquier nivel resultante de dividir una pieza de software en componentes (denominados 'Nivel 1', por ejemplo), después dividir dichos componentes en subcomponentes ('Nivel 2'), y después dividir estos subcomponentes en sub-subcomponentes ('Nivel 3'), etc.</p> <p>NOTA 1: No debe confundirse con 'nivel de granularidad'.</p> <p>NOTA 2: Las mediciones de tamaño de componentes de una pieza de software sólo pueden ser comparadas con componentes semejantes, por ejemplo, componentes del mismo nivel de descomposición.</p>

La Nota 2 en la definición anterior es importante porque el tamaño de piezas de software en diferentes niveles de descomposición no pueden simplemente sumarse sin tener en cuenta las reglas de agregación de la sección 4.3.1. Además, como consecuencia de ello, el rendimiento (por ejemplo, la productividad = tamaño / esfuerzo) de los proyectos para el desarrollo de diferentes piezas de software sólo puede ser comparado con seguridad si todas las piezas de software están en el mismo nivel de descomposición.

EJEMPLO: La Guía para 'Patrones de Estrategia de Medición' [5] reconoce tres niveles estándar de descomposición: 'Aplicación Total', 'Componente Mayor' y 'Componente Menor'. Véase el EJEMPLO 2 APLICACIÓN DE NEGOCIO en el apartado 2.2.2 de este Manual de Medición, donde los tres niveles se muestran en la Figura 2.4.

2.2.4 Definiendo el alcance de la medición: Resumen

La determinación del alcance de una medición puede implicar algo más que simplemente decidir qué funcionalidad se debe incluir en la medición. La decisión también puede implicar la consideración de la(s) capa(s) en las que reside y el nivel de descomposición del software que va a medirse, todo depende del propósito de la medición.

2.3 Identificando usuarios funcionales y el almacenamiento persistente

2.3.1 El tamaño funcional puede variar con los usuarios funcionales

Los diferentes tipos de usuarios de una 'cosa' pueden 'ver' una funcionalidad diferente y por lo tanto pueden obtener diferentes tamaños de la 'cosa'. En el caso del software, diferentes (tipos de) usuarios

funcionales pueden requerir (a través de los FUR) diferente funcionalidad y por lo tanto los tamaños funcionales variará con la elección de los usuarios funcionales.

Un 'usuario' se define, en efecto⁴, como 'cualquier cosa que interactúa con el software que se está midiendo'. Esta definición es demasiado amplia para las necesidades del método COSMIC. Para el método COSMIC, la elección del usuario (o usuarios) está determinado por los Requisitos Funcionales del 'Usuario' que debe medirse. Este (tipo de) usuario, conocido como el 'usuario funcional', se define de la siguiente manera.

DEFINICIÓN – Usuario Funcional
Un (tipo de) de usuario que es un emisor y/o un destinatario de los datos en los Requisitos Funcionales de Usuario de una pieza de software.

En el método COSMIC es esencial distinguir a los usuarios funcionales de una pieza del software que debe ser medido de entre todos sus posibles usuarios. Los FUR del software normalmente identifican a los usuarios funcionales que son los emisores de datos y/o los destinatarios de los datos hacia y desde el software, respectivamente.

EJEMPLO 1: Considere una aplicación empresarial; Sus usuarios funcionales normalmente incluirían humanos y otras aplicaciones semejantes con las que la aplicación interactúa. Para una aplicación en tiempo real, los usuarios funcionales normalmente serían dispositivos hardware u otros software semejantes que interactúen con ella.

Sin embargo, el conjunto total de 'usuarios', es decir, incluida 'cualquier cosa que interactúa con el software', debe incluir el sistema operativo. Pero los FUR de cualquier aplicación nunca incluirían el sistema operativo como un usuario. Cualquier requisito que el sistema operativo pueda imponer a una aplicación será común para todas las aplicaciones, por lo general, se encargarán de ella el compilador o intérprete, y son invisibles para los usuarios funcionales reales de la aplicación y por lo tanto no aparecen en los FUR. En la práctica de medición de tamaño funcional, un sistema operativo nunca sería considerado como un usuario funcional de una aplicación. Pero no siempre se da el caso de que los usuarios funcionales sean obvios.

EJEMPLO 2: Tomemos el ejemplo de la aplicación software de un teléfono móvil. A pesar de que hemos eliminado el sistema operativo del teléfono móvil como un posible usuario funcional de la aplicación, los 'usuarios' podrían ser tanto (a) los seres humanos que pulsan las teclas, o (b) los dispositivos de hardware (por ejemplo, la pantalla, teclado, etc.) y (c) las aplicaciones semejantes que interactúan directamente con la aplicación del teléfono. El usuario ser humano, por ejemplo, verá sólo un subconjunto de toda la funcionalidad de la que debe ser provista a fin de que la aplicación del teléfono móvil funcione. Por lo tanto, estos dos tipos de usuarios verán diferente funcionalidad, el tamaño de los FUR para los usuarios seres humanos será menor que el tamaño de los FUR que deben desarrollarse para hacer la aplicación de teléfono funcione.⁵

⁴ Ver el glosario para la definición, tomada de ISO/IEC 14143/1:2007

⁵ Toivonen, por ejemplo, la funcionalidad comparada de los teléfonos móviles disponibles para usuarios humanos en 'Defining measures for memory efficiency of the software in mobile terminals', International Workshop on Software Measurement, Magdeburg, Germany, October 2002.

REGLAS – Usuarios Funcionales

- a) Los usuarios funcionales de una pieza de software a ser medida, deben ser derivados del propósito de la medición
- b) Para los usuarios funcionales que son funcionalmente idénticos según los FUR, se identifica un solo tipo de usuario funcional.
- c) Cuando el propósito de una medición de una pieza de software está relacionado con el esfuerzo para desarrollar o modificar el software, entonces los usuarios funcionales deben ser todos aquellos emisores y/o receptores de datos hacia/desde la nueva funcionalidad o la modificada, como es requerido por sus FUR.

APLICACIÓN DE NEGOCIO EJEMPLO 3 que ilustra la regla b): En un sistema de pedidos un número de empleados (usuarios funcionales) mantienen los datos del pedido. Identificar un solo tipo de usuario funcional 'empleado'.

TIEMPO REAL EJEMPLO 4 que ilustra la regla b): Cada rueda de un coche tiene un sensor que obtiene la presión de aire. Si la presión es demasiado baja o demasiado alta - los valores están en el software - el software activa el aviso correspondiente LED rojo (s) en el tablero de instrumentos. Los usuarios funcionales son los cuatro sensores y los cuatro LEDs. Como tanto los cuatro sensores y los cuatro LEDs son funcionalmente idénticos, identificar a un usuario funcional de tipo 'sensor' de los sensores y de un tipo de usuario funcional 'LED' para el LED.

Tenga en cuenta que si hubiera un solo LED de advertencia en el panel de control que indica que al menos uno de los neumáticos tiene un fallo de la presión, los tipos de usuarios funcionales son los mismos: un tipo de usuario funcional 'sensor' para los cuatro sensores y un tipo de usuario funcional 'LED' para el LED.

Después de haber identificado los usuarios funcionales, es entonces fácil de identificar la frontera. La Frontera se encuentra entre la pieza de software que se está midiendo y sus usuarios funcionales. Ignoramos cualquier otro hardware o software en ese espacio intermedio⁶.

DEFINICIÓN – Frontera

Una interfaz conceptual entre el software que está siendo medido y sus usuarios funcionales.

NOTA: Se desprende de la definición que hay una frontera entre dos piezas de software en las mismas o diferentes capas que intercambian datos donde una pieza de software es un usuario funcional de la otra, y/o viceversa.

NOTA: Esta definición de 'frontera' se toma de ISO / IEC 14143/1: 2007, modificado por la adición de 'funcional' para calificar al 'usuario'. Para evitar ambigüedades, tenga en cuenta que el límite no se debe confundir con cualquier línea que podrían ser dibujada alrededor de algún tipo de software que debiera medirse para definir el alcance de medición. El límite no se utiliza para definir el alcance de una medición.

2.3.2 Almacén persistente

⁶ De hecho, si el medidor tiene que examinar los FUR para identificar a los emisores y posibles receptores de datos, la frontera debe de haber sido identificada.

DEFINICIÓN – Almacén persistente
<p>Un almacén persistente es un almacén que permite a un proceso funcional almacenar un grupo de datos más allá de la vida del proceso funcional y/o del cual un proceso funcional puede recuperar un grupo de datos almacenado por otro proceso funcional, o almacenado por una ocurrencia anterior del mismo proceso funcional, o almacenado por otro proceso.</p> <p>NOTA 1: En el modelo COSMIC, el almacén persistente es un concepto que existe solamente dentro de la frontera del software que se está midiendo, no es considerado como un usuario funcional del software que se está midiendo.</p> <p>NOTA 2: Un ejemplo de 'otro proceso' sería la manufactura de memoria de solo lectura.</p>

(Para la definición de un 'proceso funcional', véase la sección 3.2)

2.3.3 Diagramas de Contexto

Puede ser muy útil cuando se define un alcance de medición y los usuarios funcionales dibujar un 'diagrama de contexto' para el software que se está midiendo. En este y otras directrices de COSMIC, se utilizan los diagramas de contexto para mostrar el alcance de una pieza de software que se va a medir dentro de su contexto de usuarios funcionales (seres humanos, los dispositivos de hardware u otro software) y los movimientos de datos entre ellos. (Los diagramas de contexto por lo general también muestran almacenamiento persistente, si es pertinente.)

Un diagrama de contexto es efectivamente una instancia de un patrón de medición (ver sección 2.0) aplicada al software que se mide. Los símbolos clave utilizados en los diagramas de contexto son como se indica en la figura 2.5:

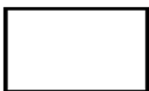

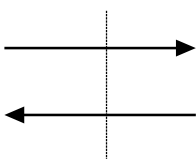
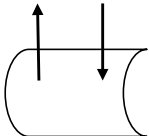
Símbolo	Interpretación
	La pieza de software que se desea medir (caja con contorno continuo y grueso), es decir, la definición de un alcance de medición.
	Cualquier usuario funcional del software que se desea medir.
	Las flechas representan <u>todos</u> los movimientos de los datos que cruzan una frontera (la línea de puntos) entre un usuario funcional y el software que se mide.
	Las flechas representan <u>todos</u> los movimientos de datos entre el software que se está midiendo y 'almacen persistente'. (El símbolo de diagrama de flujo estándar para 'almacenamiento de datos' hace hincapié en que el almacenamiento persistente es un concepto abstracto. El uso de este símbolo indica que el software no interactúa directamente con el almacenamiento de hardware físico.)

Figura 2.5 - Símbolos clave utilizados en los diagramas de contexto

APLICACIÓN DE NEGOCIO EJEMPLO: La Figura 2.6 muestra el diagrama de contexto para el software de tipo cliente/servidor del paquete implementado como en el ejemplo mostrado en la figura 2.1 de la sección 2.2.1, a ser medido como un 'todo', es decir, el hecho de que el paquete de aplicación tiene dos componentes (cliente y servidor) es ignorado para la medición del paquete.

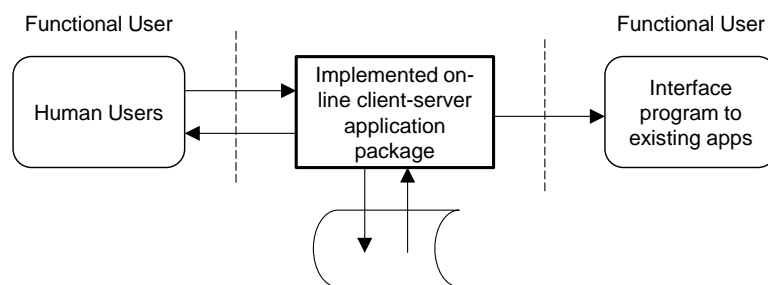


Figura 2.6 - Diagrama de contexto para la aplicación cliente-servidor de la sección 2.2.1

TIEMPO REAL EJEMPLO: La figura 2.7 muestra el diagrama de contexto para un sistema de software embebido simple de alarma para intrusos (tomado de la 'Guía COSMIC para el dimensionamiento de software en tiempo real' [4], la versión 1.1, sección 4.3)

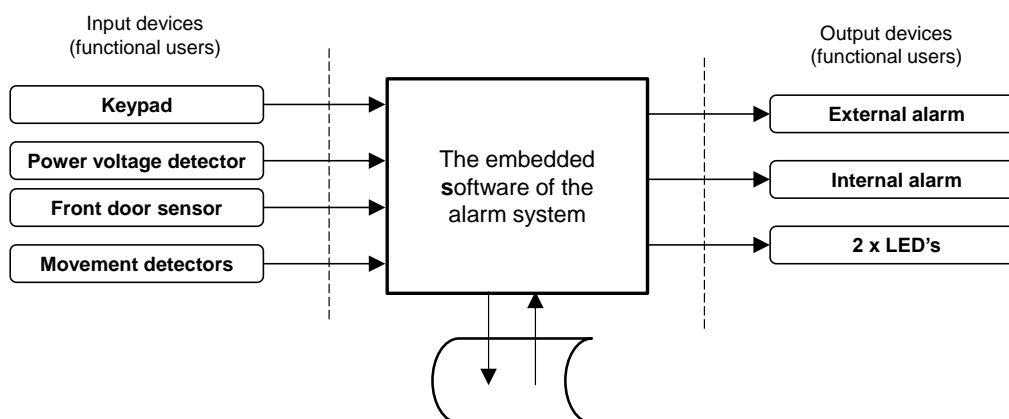


Figura 2.7 - Diagrama de Contexto para el software embebido de un sistema de alarma de intrusos

2.4 Identificando el nivel de granularidad

2.4.1 La necesidad de un nivel de granularidad estándar

En las etapas iniciales de un proyecto de desarrollo de software, los requisitos funcionales de los usuarios (FUR) se especifican en 'alto nivel', es decir, se obtiene un esbozo, o con pocos detalles. A medida que el proyecto progresa, los FUR, son refinados, (por ejemplo, a través de las versiones 1, 2, 3, etc.), revelando más detalle a un 'menor nivel'. Estos diferentes grados de detalle de los requisitos actuales (y por tanto derivados de los FUR), son conocidos como diferentes 'niveles de granularidad'. (Véase también la sección 2.4.2 para otros términos que pueden confundirse con el concepto de 'nivel de granularidad' como es definido aquí)

DEFINICIÓN – Nivel de granularidad

Cualquier nivel de expansión de la descripción de una pieza de software (por ejemplo, una especificación de requisitos, o una descripción de la estructura de la pieza de software) de tal manera que a cada aumento del nivel de expansión, la descripción de la funcionalidad de la pieza de software se encuentra en un nivel de detalle mayor y uniforme.

NOTA: Los medidores deben ser conscientes de que cuando los requisitos van apareciendo a principios de la vida de un proyecto de software, en cualquier momento diferentes partes de la funcionalidad necesaria del software normalmente se habrán documentado en los diferentes niveles de granularidad.

En la mayoría de las actividades de desarrollo de productos, los planos son dibujados a escalas estándares, y es fácil de traducir dimensiones medidas en un plano a los de otro plano con una escala diferente. Por el contrario no hay escalas estándares para los distintos niveles de granularidad en los que el software puede ser especificado, por lo que puede ser difícil asegurar que dos especificaciones de FUR están al mismo nivel de granularidad. Sin un acuerdo sobre un cierto nivel estándar de granularidad en el cual medir (o al que las mediciones deben ser escalados) es imposible saber con certeza que dos mediciones de tamaño funcional pueden compararse.

Para ilustrar los problemas aún más, considere esta analogía. Un conjunto de mapas de carreteras revela los detalles de una red nacional de carreteras en tres niveles de granularidad:

- Mapa A muestra sólo las autopistas y carreteras principales;
- Mapa B muestra todas las autopistas, carreteras principales y secundarias (como en un atlas para los automovilistas);
- Mapa C muestra todas las carreteras secundarias con sus nombres (como en un conjunto de mapas de carreteras del distrito local).

Si no reconocemos el fenómeno de los diferentes niveles de granularidad, parece que se exponen tres mapas diferentes de la red de carreteras de la nación. Por supuesto, con mapas de carreteras todo el mundo reconoce los diferentes niveles de detalle y hay escalas estándar para interpretar el tamaño de la red descrita en cualquier nivel. El concepto abstracto de nivel de granularidad se esconde detrás de las escalas de estos mapas distintos

Para la medición del software, sólo hay un nivel de granularidad estándar que es posible definir sin ambigüedades. Es el nivel de granularidad en el cual se han identificado los distintos procesos funcionales y se han definido sus movimientos de datos. Las mediciones deben hacerse a este nivel o a escala de este nivel siempre que sea posible⁷.

2.4.2 Aclaración del nivel de granularidad

Antes de continuar, es importante asegurar que no haya malentendidos sobre el significado de 'nivel de granularidad' en el método COSMIC. Detallar los FUR implica la ampliación de la descripción del software de un nivel de granularidad 'superior' a uno 'inferior' revelando más detalles pero *sin modificar su alcance*. Este proceso no debe confundirse con alguno de los siguientes.

- Detallar un software con el fin de revelar sus componentes, subcomponentes, etc. (en diferentes 'niveles de descomposición'- véase la sección 2.2.3 anterior). Tal definición de detalle puede ser necesario si el propósito de medición requiere que el alcance general de la medición sea subdividida siguiendo la estructura física del software.
- La evolución de la descripción de algún tipo de software a medida que avanza a través de su ciclo de desarrollo, por ejemplo, de los requisitos al diseño lógico, al diseño físico, etc. Cualquiera que sea la etapa en el desarrollo de algún tipo de software, sólo estamos interesados en su FUR para fines de medición.

El concepto de 'nivel de granularidad' se aplica únicamente a los requisitos de los usuarios funcionales de software (FUR).

2.4.3 El nivel de granularidad estándar

Mediciones de tamaño funcional COSMIC exactos requieren que el FUR a ser medido existe a un nivel de granularidad en el que los procesos funcionales y sus movimientos de datos pueden ser identificados. El 'nivel de granularidad de proceso funcional' se define como sigue⁸.

⁷ El tema de escalamiento de las mediciones de un nivel de granularidad a otro está tratado en el documento 'Temas Avanzados y Relacionados' [6]. A 'Guía para la aproximación de la medición de tamaño funcional COSMIC' [12] está en desarrollo que eventualmente sustituirá al documento existente.

⁸ La razón del nombre 'nivel de granularidad de proceso funcional' es que este es el nivel en el que se identifican los procesos funcionales - véase la sección 3.2 para una discusión más detallada de los procesos funcionales.

DEFINICIÓN - Nivel de Granularidad de un Proceso Funcional

Un nivel de granularidad de la descripción de una pieza de software en el que los usuarios funcionales:

- Son seres humanos individuales o dispositivos de ingeniería o elementos de software (y no grupos de estos) Y
- detectan ocurrencias únicas de eventos a los que la aplicación software debe responder (y no cualquier nivel en el cual se han definido grupos de eventos)

NOTA 1: En la práctica, la documentación de software y por lo tanto los Requisitos Funcionales de Usuario a menudo describe la funcionalidad en diferentes niveles de granularidad, sobre todo cuando la documentación aún está en desarrollo.

NOTA 2: 'Grupos de estos' (usuarios funcionales) podrían ser, por ejemplo, un 'departamento' cuyos miembros manejan muchos tipos de procesos funcionales; o un 'panel de control' que tiene muchos tipos de instrumentos; o 'sistemas centrales'.

NOTA 3: Un grupo de eventos puede, por ejemplo, indicarse en una especificación de FUR en un alto nivel de granularidad por un flujo de entrada a un sistema de software de contabilidad, etiquetados como 'transacciones de venta', o por un flujo de entrada a un sistema de software de aeronáutica, etiquetado como 'comandos del piloto'.

Con esta definición, podemos ahora definir las siguientes reglas y una recomendación.

REGLAS - Nivel de granularidad de un proceso funcional

- a) Una medición precisa de tamaño funcional de una pieza de software requiere que sus FUR sean conocidos a nivel de granularidad en la que se pueden identificar sus procesos funcionales y sub-procesos de movimiento de datos.
- b) Si algunos requisitos deben medirse antes de que se hayan definido con suficiente detalle para una medición precisa, los requisitos se pueden medir utilizando un enfoque de aproximación. Estos enfoques definen cómo los requisitos pueden ser medidos en niveles de granularidad superiores. Los factores de escala se aplican entonces a las mediciones realizadas a niveles de granularidad más altos para producir un tamaño aproximado al nivel de granularidad de procesos funcionales y sus subprocesos de movimientos de datos. Consulte la 'Guía para la aproximación de la medición de tamaño funcional COSMIC' [6].

Además de las reglas, COSMIC recomienda que el nivel de granularidad al cual un proceso funcional y sus subprocesos de movimientos de datos son conocidos, debe ser el estándar al que se requiere medir el tamaño funcional y ser utilizado por los proveedores de servicios de evaluaciones comparativas y de herramientas de software diseñados para soportar o utilizar mediciones de tamaño funcional, por ejemplo para estimar el esfuerzo de un proyecto.

APLICACIÓN DE NEGOCIO EJEMPLO: El ejemplo del ámbito de software de negocio, será de un conocido sistema de comprar de productos a través de Internet, que llamaremos el sistema 'Aplicación de Órdenes Everest'. La descripción que se muestra a continuación está muy simplificada para los propósitos de esta ilustración de los niveles de granularidad.

Si quisiéramos medir esta aplicación, podemos asumir el propósito de la medición es determinar el tamaño funcional de la parte de la aplicación disponible para los clientes seres humanos (como los 'funcionales'). Entonces tendríamos definir el alcance de la medida como 'las partes de la aplicación del Everest accesibles a los clientes para encargar productos a través de Internet'. Tenga en cuenta, sin embargo, que el propósito de este ejemplo es para ilustrar distintos niveles de granularidad. Por

lo tanto, vamos a explorar sólo algunas partes de la funcionalidad total del sistema suficientes para entender este concepto de niveles de granularidad.

Al más alto 'Nivel 1 (Función principal)' de esta parte de la aplicación una especificación de requisitos de la Aplicación Everest sería un resumen tan simple como lo siguiente.

“La Aplicación Everest deberá permitir a los clientes informarse sobre, seleccionar, pedir, pagar y obtener la entrega de cualquier elemento de la gama de productos Everest, incluidos los productos disponibles desde terceros proveedores.”

Detallando esta especificación de requisitos de alto nivel nos encontramos con que en el siguiente nivel inferior 2, el sistema consiste en cuatro sub-sistemas, tal y como se muestra en Fig. 2.8 (a).

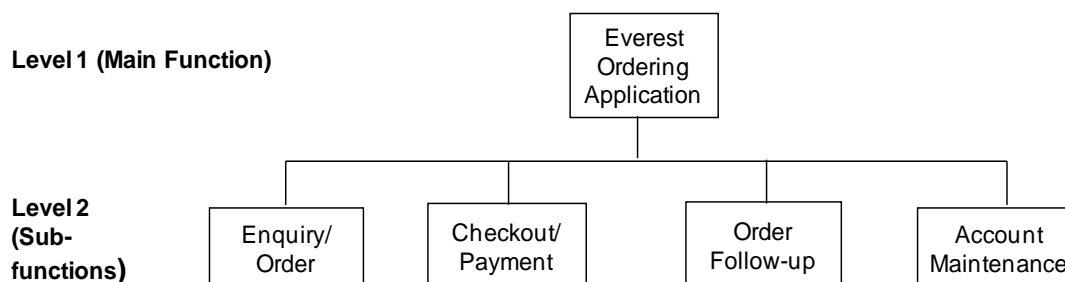


Figura 2.8 (a) - Análisis del sistema de pedidos Everest: los dos primeros niveles de granularidad

Los requisitos de los cuatro sub-sistemas son los siguientes:

- El sub-sistema de la Solicitud/Orden que permite a un cliente encontrar cualquier producto en la base de datos de Everest, así como su precio y disponibilidad y añadir cualquier producto seleccionado a una 'cesta' de compra. Este sub-sistema también promueve las ventas sugiriendo las ofertas especiales, que ofrece comentarios de los elementos seleccionados y permite preguntas generales como las condiciones de entrega, etc. Se trata de un sub-sistema muy complejo y no es analizado en más detalle por debajo del nivel 2 a efectos del presente ejemplo.
- El sub-sistema de la compra/pago que permite a un cliente comprometerse con el pedido y pagar por la mercancía de la cesta de compra.
- El sub-sistema del Seguimiento del pedido que permite a un cliente a preguntar hasta qué punto un pedido existente ha avanzado en el proceso de entrega, para mantener su pedido (por ejemplo, cambiar la dirección de entrega) y para devolver bienes no satisfactorios.
- El sub-sistema del Mantenimiento de la cuenta que permite a un cliente existente, mantener diversos detalles de su cuenta tales como domicilio, medios de pago, etc.

La figura 2.8 (b) and (c) también muestra algún detalle que se pone de manifiesto cuando detallamos los otros dos niveles del subsistema de La Compra/Pago, del Seguimiento del Pedido y el subsistema de Mantenimiento de la Cuenta. En este proceso, es importante señalar que:

- no hemos cambiado el alcance del sistema de la aplicación que habrá de medirse, y
- todos los niveles de la descripción de la aplicación de software Everest muestran la funcionalidad disponible para los clientes (como usuarios funcionales). Un cliente puede 'ver' la funcionalidad del sistema en todos estos los niveles de granularidad.

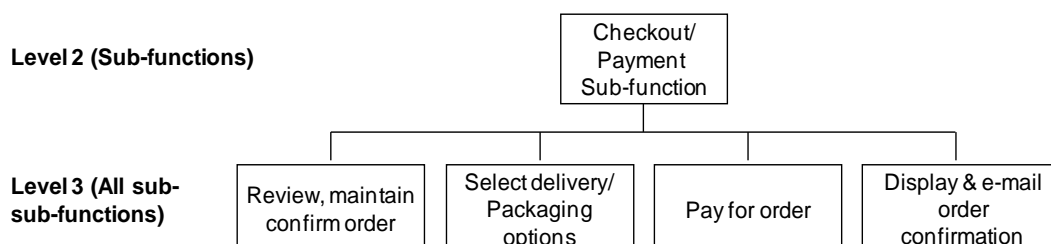


Figura 2.8 (b) - La descomposición del sub-sistema de la compra/pago

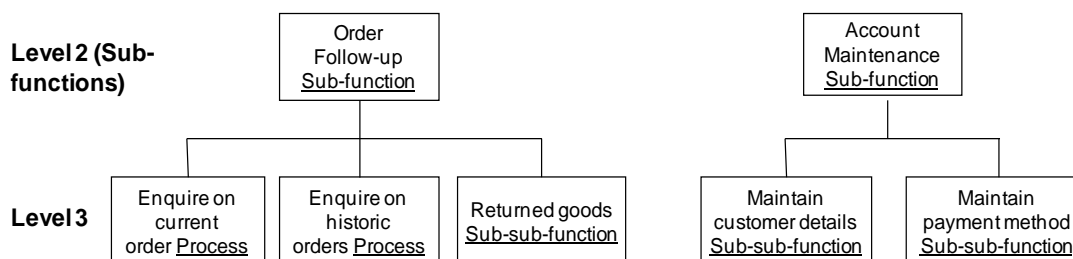


Figura 2.8 (c) - La descomposición del sub-sistema del Seguimiento y del sub-sistema del Mantenimiento de la cuenta

La Figura 2.8 (c) revela que cuando detallamos y nos acercamos en el nivel inferior 3 de este análisis particular del sub-sistema del Seguimiento, nos encontramos dos procesos funcionales individuales en el nivel 3 (para dos consultas del sub-sistema de Seguimiento del Pedido). Este ejemplo demuestra, por tanto, que cuando alguna funcionalidad es analizada en un enfoque de 'arriba hacia abajo', no se puede suponer que la funcionalidad mostrada en un 'nivel' en particular en un diagrama siempre corresponde con el mismo 'nivel de granularidad' tal y como este concepto es definido en el método COSMIC. (Esta definición exige que a cualquier nivel de granularidad la funcionalidad esté a un "nivel comparable de detalle").

Por otra parte, otros analistas podrían dibujar el diagrama de forma diferente, mostrando otras agrupaciones de funcionalidades en cada nivel del diagrama, ya que no hay una única forma 'correcta' de detallar la funcionalidad de un sistema complejo.

Teniendo en cuenta estas variaciones que inevitablemente ocurren en la práctica, un medidor debe examinar cuidadosamente los diversos niveles de un diagrama de análisis para encontrar los procesos funcionales que deben medirse. En caso de que en la práctica esto no sea posible, por ejemplo debido a que el análisis aún no ha alcanzado el nivel donde todos los procesos funcionales se han revelado, debe aplicarse la regla (b) descrita más arriba. Para ilustrar esto, vamos a examinar el caso del sub-subsistema 'Mantener Detalles del Cliente' (véase la Figura 2.8 (c) arriba), en la rama del sub-sistema de Mantenimiento de la Cuenta.

Para un medidor con experiencia, la palabra 'mantener' casi siempre sugiere un grupo de casos y por lo tanto, un grupo de procesos funcionales. Por lo tanto, podemos asumir con seguridad que este sub-subsistema de 'mantener' debe estar integrado por tres procesos funcionales, es decir, 'informe sobre detalles del cliente', 'actualización de los detalles del cliente' y 'eliminar detalles del cliente'. (Evidentemente el proceso de 'creación del cliente' también existe pero esto ocurre en otra rama del sistema, que es cuando un cliente hace un pedido por primera vez. Está fuera del ámbito de aplicación de este ejemplo).

Un medidor con experiencia debe poder estimar un tamaño de este sub-subsistema en unidades COSMIC Function Points tomando el supuesto número de procesos funcionales (tres en este caso) y multiplicando este número por el tamaño promedio de un proceso funcional. Este tamaño medio se obtendría calibrando en otras partes de este sistema o en otros sistemas comparables. Ejemplos de este proceso de calibración se dan en el documento de 'Guía para la aproximación de la medición de tamaño funcional COSMIC' [6] que también contiene otros ejemplos de enfoques para aproximar el tamaño.

Evidentemente, estos métodos de aproximación tienen sus limitaciones. Si aplicamos este enfoque al nivel 1 de la especificación de requisitos de la forma arriba señalada (El sistema "Everest" deberá permitir a los clientes informarse sobre, seleccionar, comprar, pagar y obtener la entrega de cualquier elemento de la gama de productos Everest...), podríamos identificar unos procesos funcionales. Pero un análisis más detallado podría revelar que el verdadero número de procesos funcionales de este complejo sistema debe ser mucho mayor. Esa es la razón por la que tamaños funcionales aparentan aumentar a medida que se establecen más detalles en los requisitos, incluso sin cambios en el alcance de la aplicación. Estos métodos de aproximación, por lo tanto, deben ser utilizados con sumo cuidado a los altos niveles de granularidad, cuando está disponible muy poca información.

Para un ejemplo de medición a varios niveles de granularidad y de descomposición, véase el ejemplo del sistema de telecomunicaciones en la Guía para la aproximación de la medición de tamaño funcional COSMIC [6].

2.5 Observaciones finales sobre la Fase de Estrategia de Medición

Es esencial para determinar y documentar los parámetros de la estrategia de medición a fin de garantizar que el tamaño resultante puede entenderse y utilizarse en el futuro correctamente. La gran mayoría de las mediciones de tamaño funcionales se llevan a cabo para un propósito que está relacionado con el esfuerzo de desarrollo de alguna manera, por ejemplo, para la medición de los resultados de desempeño del proyecto, o para la estimación del proyecto. En muchas de estas situaciones, un patrón de medida estándar puede utilizarse: Véase la Guía para 'Patrones de Estrategia de Medición' [5].

FASE DE REPRESENTACIÓN

3.0 Resumen del Capítulo

Este capítulo trata de la 'Representación', segunda fase del proceso, de medición mediante la definición de los conceptos clave del Modelo Genérico de Software y el proceso a seguir en la representación de los FUR del software en el modelo, con la finalidad de que los FUR se puedan medir. Estos conceptos clave del Modelo Genérico de Software son: (*en cursiva*):

- Un evento hace que un *usuario funcional* solicite un servicio de la pieza de software que se está midiendo. Tal evento se llama un '*evento desencadenante*' y el servicio solicitado un '*proceso funcional*'.
- Los procesos funcionales se componen de dos tipos de subprocesos que, o bien mueven datos ('*movimientos de datos*') o manipulan datos ('*manipulación de datos*'). Los subprocesos de manipulación de datos no se reconocen por separado, pero se considera que se explica por los movimientos de datos con los que están asociados.
- Un movimiento de datos mueve un '*grupo de datos*'. Un grupo de datos se compone de '*atributos de datos*' que todos describen un '*objeto de interés*', es decir, un objeto que es de interés en el mundo del usuario funcional relacionado.
- Hay cuatro tipos de movimientos de datos: *Entradas* y *Salidas* cada uno mueve un grupo de datos dentro y fuera de un proceso funcional a través de una frontera de/hacia un usuario funcional respectivamente. *Lectura* y *Escritura* cada uno mueve un grupo de datos entre el proceso funcional y el *almacenamiento persistente*.

Cada uno de estos conceptos se define en este capítulo, que incluye principios generales y reglas para ayudar a identificar los conceptos correctamente, así como extensos ejemplos de los conceptos para los diferentes tipos de software.

3.1 Representación de los FUR al Modelo Genérico de Software

La figura 3.0 muestra los pasos del proceso para mapear los Requisitos Funcionales de Usuario (FUR) y los artefactos de software disponibles a la forma requerida por el Modelo Genérico de Software de COSMIC.

Cada paso en este proceso es el objeto de una sección específica que se indica en la barra de título de la etapa en la Figura 3.0.

El proceso está diseñado para ser aplicable a una gama muy amplia de artefactos de software. Impulsamos a los medidores a utilizar este proceso general para derivar reglas más específicas para el uso en su entorno local para asignar los artefactos de software producidos por su método de ingeniería de software local al Modelo Genérico de Software de COSMIC. El objetivo de un proceso local específico, ilustrado con ejemplos locales, debe ser reducir la incertidumbre en la asignación y por lo tanto para mejorar la precisión y la repetibilidad de las mediciones.

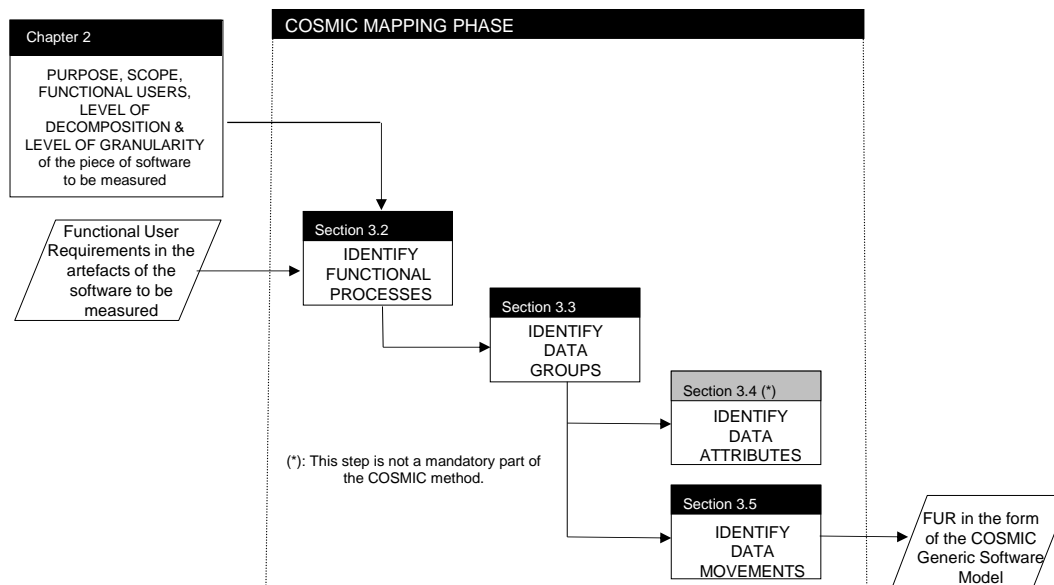


Figura 3.0 – Método general del proceso de representación COSMIC

Varias Guías que describen cómo mapear de varios análisis de datos y métodos de determinación de requisitos utilizados en diferentes ámbitos a los conceptos del método COSMIC, están disponibles. Ejemplos de ello son la ‘Guía para el dimensionamiento de aplicaciones de negocios’ [7], la ‘Guía para el dimensionamiento de aplicaciones de Data Warehouse’ [8], la ‘Guía para dimensionamiento de aplicaciones con una Arquitectura Orientada a Servicios’ [9] y la ‘Guía para el dimensionamiento de software en tiempo real’ [4]. Para aplicaciones de negocio [10] y aplicaciones en tiempo real [11] también hay guías rápidas de referencia disponibles que dan una visión general del proceso en unas pocas páginas.

El objetivo de las figuras 3.1 y 3.2 es ayudar a nuestra transición del Modelo Contextual de Software usado en la Fase de la Estrategia de Medición hacia el Modelo Genérico de Software. Las figuras se aplican para una pieza de software de aplicaciones de negocio y una pieza típica de software embebido en tiempo real, y corresponden a las figuras 2.6 y 2.7, respectivamente.

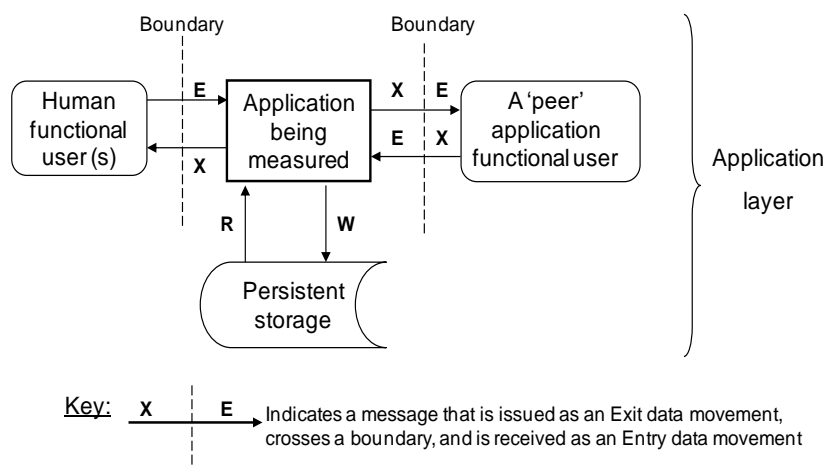


Figura 3.1 - Una aplicación de negocios con seres humanos y aplicaciones 'semejantes' como sus usuarios funcionales.

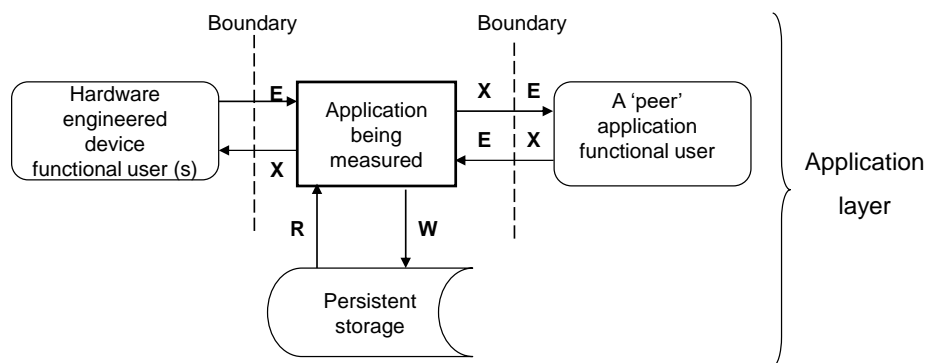


Figura 3.2 - Una aplicación de software embebido en tiempo real con varios dispositivos de hardware y una aplicación 'semejante' como sus usuarios funcionales

Del principio f) del Modelo Contextual Software (véase la sección 1.3.1), observamos que los usuarios funcionales del software que se miden son los 'remitentes y/o destinados o destinatarios de los datos hacia/desde el software, respectivamente'. El principio a) del Modelo Genérico de Software (ver sección 1.3.2) nos dice que una pieza de software 'interactúa con sus usuarios funcionales a través de una frontera y con almacenamiento persistente dentro de este límite'.

La Fig. 3.1 muestra el software de la aplicación que se está midiendo con usuarios funcionales; personas y otra aplicación semejante. La aplicación de software embebido de Figura 3.2 tiene sólo dispositivos de hardware y una aplicación semejante como sus usuarios funcionales. Las flechas que muestran los movimientos de datos están etiquetados con las siglas mostrando sus tipos (E = Entrada, X = Salida, R = Lectura, W = Escritura).

Recuerde que el 'almacenamiento persistente' se refiere a cualquier almacenamiento lógico que el software que se está midiendo necesite para acceder a través de movimientos de datos de Lectura o Escritura; que no implica ningún tipo de almacenamiento físico.

3.2 Identificación de los procesos funcionales

Este paso consiste en identificar el conjunto de procesos funcionales de la aplicación software que se va a medir a partir de sus Requisitos Funcionales de Usuario.

3.2.1 Definiciones

DEFINICIÓN – Evento
Algo que sucede.

DEFINICIÓN – Evento desencadenante
<p>Un evento (algo que ocurre) que genera que un usuario funcional del componente de software desencadene ("trigger") uno o más procesos funcionales. En un conjunto de requisitos funcionales de usuario, cada evento que causa que un usuario funcional desencadene un proceso funcional</p> <p>Un evento, reconocido en los Requisitos Funcionales de Usuario del software que se está midiendo, que hace que uno o más usuarios funcionales de este software genere uno o más grupos de datos, cada uno de los cuales posteriormente serán movidos por una Entrada desencadenante. Un evento desencadenante no puede ser subdividido y sucede o no sucede.</p> <p>NOTA: los eventos de reloj y temporización pueden ser eventos desencadenantes.</p>

DEFINICIÓN – Proceso funcional
<p>a) Un conjunto de movimientos de datos, que representa una parte elemental de los Requisitos Funcionales de Usuario para el software que se está midiendo, que es único dentro de estos FUR y que se puede ser definido de manera independiente de cualquier otro proceso funcional en estos FUR.</p> <p>b) Un proceso funcional puede tener sólo una Entrada desencadenante. Cada proceso funcional inicia el procesamiento con la recepción de un grupo de datos movidos por el movimiento de datos de Entrada desencadenante del proceso funcional.</p> <p>c) El conjunto de todos los movimientos de datos de un proceso funcional es el conjunto que se necesita para cumplir con los FUR para todas las posibles respuestas a la Entrada desencadenante.</p> <p>NOTA 1: Cuando se implementa, es una <i>ocurrencia</i> de un proceso funcional que comienza <i>ejecutando</i> con la recepción de una <i>ocurrencia</i> de un grupo de datos movido por una <i>ocurrencia</i> de una Entrada desencadenante.</p> <p>NOTA 2: El FUR para un proceso funcional puede requerir una o más Entradas distintas, además de la entrada desencadenante.</p> <p>NOTA 3: Si un usuario funcional envía un grupo de datos con errores, por ejemplo, porque un sensor funciona mal o una orden introducida por una persona tiene errores, por lo general es la tarea del proceso funcional para determinar si el evento realmente ocurrió y/o si los datos introducidos son realmente válidos y cómo responder.</p>

DEFINICIÓN – Entrada desencadenante
El movimiento de Entrada de datos de un proceso funcional que mueve un grupo de datos generados por un usuario funcional que el proceso funcional necesita para iniciar el procesamiento.

La relación entre un evento desencadenante, el usuario funcional y el movimiento de datos de Entrada que desencadena un proceso funcional que se está midiendo se esquematiza en la figura 3.3 de abajo. La interpretación de este diagrama es: *un evento desencadenante ocasiona que usuario funcional genere un grupo de datos que se mueve mediante una Entrada desencadenante de un proceso funcional para iniciar el proceso funcional.*

NOTA: Para facilitar la lectura, a menudo omitimos la referencia al grupo de datos y declarar que un usuario funcional inicia una Entrada desencadenante que inicia un proceso funcional, o incluso más simple que un usuario funcional inicia un proceso funcional.

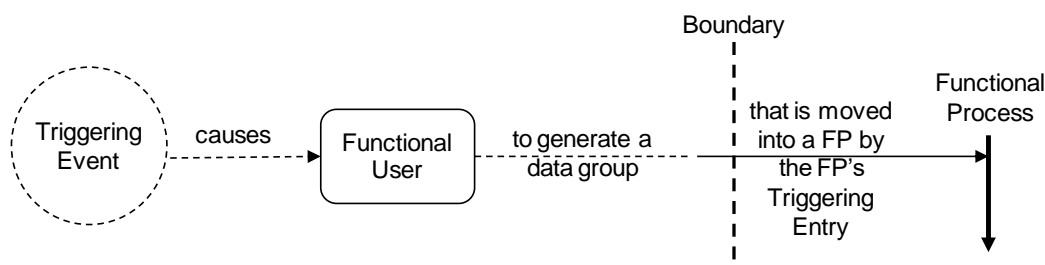


Figura 3.3 – Relaciones entre un evento desencadenante, un usuario funcional y un proceso funcional

Todas las relaciones entre los conceptos en Figura 3.3 (el evento desencadenante / el usuario funcional / Entrada desencadenante/ proceso funcional) pueden ser uno a muchos, muchos a uno, o muchos a muchos, con una excepción. La excepción es que el grupo de datos movido por cualquier Entrada

desencadenante puede iniciar sólo un proceso funcional - véase la regla b) para un proceso funcional. Algunos ejemplos de posibles cardinalidades:

- Un evento desencadenante puede ser detectado por muchos usuarios funcionales, por ejemplo, un terremoto es detectado por muchos sensores;
- Los usuarios funcionales humanos de una aplicación de negocios pueden detectar muchos tipos de eventos. Cuando un usuario humano decide el evento debe ser notificado a la aplicación, es decir, se trata de un evento desencadenante, el usuario inicia una entrada desencadenante. (Un ser humano puede incluso 'generar' efectivamente un evento cuando decide realizar una consulta.) Del mismo modo, una aplicación de software A que es un usuario funcional de otra aplicación B (que se mide), puede 'llamar' la aplicación B para diversos propósitos. Le corresponde a la aplicación A la generación de un evento desencadenante separado para cada llamada (- tipo).
- Un usuario funcional de hardware puede iniciar más de una Entrada desencadenante en la detección de un evento desencadenante en ciertos tipos de software crítico de seguridad;
- Algunos procesos funcionales pueden ser iniciadas por diferentes usuarios funcionales, por ejemplo, un componente de software reutilizable que se puede llamar por cualquiera de sus usuarios funcionales.

En la práctica, las cardinalidades a lo largo de todas las cadenas para el software que se mide (es decir, que describen los eventos específicos que causan los usuarios funcionales específicas para iniciar procesos funcionales específicos) será limitada por los FUR del software. Para una discusión más completa de las cardinalidades a lo largo de la cadena de la Figura 3.3 y para más ejemplos, consulte el Apéndice C de este Manual de Medición.

El(los) grupo(s) de dato(s) que un usuario funcional generará como resultado de un evento de activación depende del FUR del proceso funcional que procesará los datos, como se ilustra mediante los siguientes ejemplos.

APLICACIÓN DE NEGOCIO EJEMPLO 1: Un proceso funcional de un sistema de software de personal puede ser iniciado por la entrada desencadenante que mueve un grupo de datos que describe un nuevo empleado. El grupo de datos es generado por un usuario funcional humano del software que introduce los datos.

TIEMPO REAL EJEMPLO 2: Un proceso funcional de un sistema de software de tiempo real puede ser iniciado por su Entrada desencadenante informando al proceso funcional que un reloj (usuario funcional) ha marcado. El grupo de datos que se movió contiene datos (la marca de tiempo, tal vez a través de un solo bit) que informa únicamente de que ha ocurrido un evento.

TIEMPO REAL EJEMPLO 3: Un proceso funcional de un sistema de software de detección de incendios en tiempo real industrial puede ser iniciado por su Entrada desencadenante iniciada por un detector de humo específico (usuario funcional). El grupo de datos generado por el detector transmite la información 'humo detectado' (se ha producido un evento) e incluye el ID del detector (es decir, datos que se puede utilizar para determinar dónde ocurrió el evento).

TIEMPO REAL EJEMPLO 4: Un lector de código de barras (un usuario funcional) en una caja del supermercado inicia un análisis cuando un código de barras aparece en su ventana (el evento desencadenante). El lector genera un grupo de datos, que comprende una imagen del código de barras que es introducida en el software de pagar. La imagen del grupo de datos es movido por una Entrada desencadenante a su proceso funcional. Este último suma el costo del producto a la factura del cliente, si el código es válido, suena un pitido para informar al cliente que el producto ha sido aceptado, y registra la venta, etc., etc.

3.2.2 La aproximación para identificar procesos funcionales

La aproximación para identificar procesos funcionales depende de los artefactos de software que están disponibles para el medidor, los cuales dependen del momento del ciclo de vida del software cuando se requiere la medida y de los métodos de análisis, diseño y desarrollo del software en uso. Puesto que éstos últimos varían enormemente, este Manual de Medición puede proporcionar solamente un proceso general para identificar procesos funcionales

El proceso de identificación de los procesos funcionales, después de que los usuarios funcionales se han identificado y dados los FUR para el software que se está midiendo sigue la cadena de la Figura 3.3, que es:

1. Identificar en el mundo de los usuarios funcionales los eventos por separado a los que el software que se está midiendo debe responder- los 'eventos desencadenantes' (Los eventos desencadenantes se pueden identificar en diagramas de estado y en los diagramas del ciclo de vida de una entidad, ya que algunas transiciones de estado y transiciones del ciclo de vida de una entidad corresponden al desencadenamiento de eventos a los que el software debe reaccionar.);
2. Identificar qué usuario(s) funcional(es) del software puede responder a cada evento desencadenador;
3. Identifique la entrada (o entradas) desencadenante(s) que cada usuario funcional puede iniciar en respuesta al evento;
4. Identificar el proceso funcional iniciado por cada entrada desencadenante.

Utilice las siguientes reglas para comprobar que los procesos funcionales candidatos han sido adecuadamente identificados.

REGLAS – Proceso Funcional
<p>a) Un proceso funcional pertenecerá íntegramente a la alcance de la medición de una pieza de software en una, y sólo una, capa.</p> <p>b) Cualquiera Entrada desencadenante de una pieza de software que se está midiendo puede iniciar sólo un proceso funcional en ese software.</p> <p>c) Un proceso funcional comprenderá al menos dos movimientos de datos, una Entrada y además una Salida o una Escritura. No hay límite superior para el número de movimientos de datos en un proceso funcional.</p> <p>d) Un proceso funcional en ejecución se debe considerar terminado cuando se ha satisfecho su FUR para la respuesta a su Entrada desencadenante. Una pausa durante la tramitación por razones técnicas no se considerará como la terminación del proceso funcional.</p>

3.2.3 Eventos disparadores y procesos funcionales en el ámbito de aplicaciones de negocio

- a) Eventos desencadenantes de una aplicación de negocio on-line normalmente ocurren en el mundo real de los usuarios funcionales humanos de la aplicación. El usuario humano comunica la incidencia de un evento a un proceso funcional introduciendo datos sobre el evento.

APLICACIÓN DE NEGOCIO EJEMPLO 1: En una compañía, se recibe una orden (evento desencadenante), causando que un empleado (usuario funcional) introduzca los datos de la orden (Entrada desencadenante que comunica datos sobre el objeto de interés 'orden'), como el primer movimiento de datos del proceso funcional "entrada de orden"

- b) Eventos (-Tipos) desencadenantes separados y por lo tanto procesos funcionales (-Tipos) separados debe distinguirse en los siguientes casos:

Cuando un usuario funcional humano toma decisiones fuera del software sobre 'qué hacer después' que son independientes en el tiempo y que requieren respuestas separadas del software, cada decisión por separado es un evento desencadenante separado para el cual el software debe proporcionar un proceso funcional distinto.

APLICACIÓN DE NEGOCIO EJEMPLO 2: Un usuario funcional introduce una orden de un cliente para un artículo de equipo industrial sofisticado y más tarde confirma la aceptación del pedido al cliente. Entre introducir la orden y aceptarla, el usuario puede realizar consultas sobre si la nueva orden puede ser entregado en la fecha de entrega solicitada, y sobre la solvencia del cliente, etc. Aunque la aceptación de una orden debe seguir la entrada de un pedido, en este caso el usuario

debe tomar una decisión separada de aceptar el pedido. Esto indica procesos funcionales separados para la entrada de pedidos y para la aceptación del pedido (y para cada una de las consultas).

Cuando las responsabilidades para las actividades son separadas.

APLICACIÓN DE NEGOCIO EJEMPLO 3: En un sistema de personal, donde la responsabilidad de mantener los datos personales básicos se separa de la responsabilidad de mantener los datos de nómina indicando usuarios funcionales separados cada uno con sus propios procesos funcionales separados.

- c) Algunas veces, para una aplicación A podría haber una aplicación semejante B que necesite enviar datos a u obtener datos de la aplicación A. En este caso, la aplicación B desencadena un proceso funcional de la aplicación A cuando necesita enviar datos u obtener datos de esta, entonces la aplicación B es un usuario funcional de la aplicación A.

APLICACIÓN DE NEGOCIO EJEMPLO 4: Supóngase que al recibir una orden en el ejemplo (a) en la aplicación que procesa la orden se requiere que se envíen detalles del cliente a una aplicación central de registro de clientes, lo cual se está midiendo. Ahora la aplicación que procesa la orden se ha convertido en un usuario funcional de la aplicación central. La aplicación de procesamiento de órdenes, después de haber recibido datos sobre un nuevo cliente, genera el grupo de datos de clientes que se envía a la aplicación central de registro de clientes central que desencadena un proceso funcional para almacenar estos datos.

- d) No hay diferencia en el principio para el análisis de un proceso funcional si se requiere un procesamiento on-line o un procesamiento por lotes. Por definición, todos los datos que han sido introducidos como entrada para el procesamiento por lotes deben ser almacenados temporalmente en algún lugar antes de que el proceso puede comenzar. Ver Figura 3.4. (nótese que se hace la distinción de datos de entrada - todas las entradas - a partir de datos persistentes que también podrían necesitar ser leídas o escritas mediante el proceso por lotes). Al medir el software de procesamiento por lotes, los datos de entrada almacenados temporalmente, deben ser analizados en la misma forma que si se está ingresando directamente a la aplicación. Estos datos de entrada no es 'de datos persistentes'.

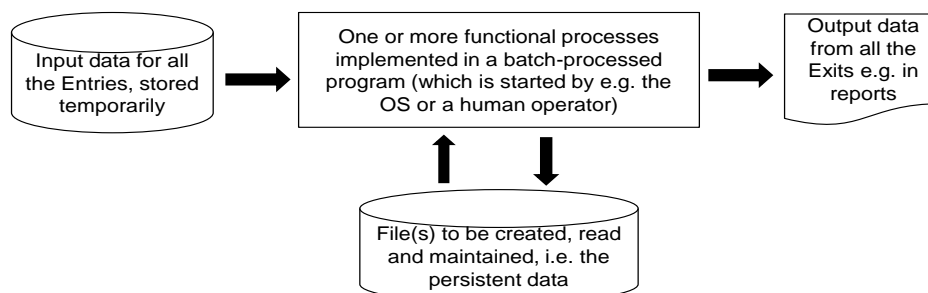


Figura 3.4 - Un 'trabajo' por procesamiento por lotes, implementación de un conjunto de procesos funcionales

NOTA: El requisito de que algunos datos de entrada sean procesados por lotes es un requisito no funcional (NFR). El efecto de esta NFR es que los datos de entrada deben estar disponibles (como un 'lote') para la entrada a la aplicación de procesamiento por lote. Cómo sucede eso en la práctica no se refiere al análisis de los FUR del software de procesamiento por lotes.

Tenga en cuenta también que cada proceso funcional (- tipo) para ser procesado en un 'trabajo' de procesamiento por lote debe ser analizado en su totalidad, independientemente de cualquier otro proceso funcional en el mismo trabajo. Cada proceso funcional en el trabajo debe tener su propia entrada desencadenante.

APLICACIÓN DE NEGOCIO 5: Supongamos que las órdenes en el ejemplo (a) anterior se introducen mediante un proceso de 'off-line' de alguna manera, por ejemplo, por escaneo óptico de documentos en papel y se almacenan temporalmente para el procesamiento por lotes automático. ¿Cómo es analizado este proceso funcional de entrada de órdenes si se va a procesar por lotes? El usuario funcional es el ser humano que hace que los datos de la orden a introducirse fuera de línea estén listos para ser procesado por lotes; la Entrada desencadenante del proceso funcional que va a procesar las órdenes por lotes es el movimiento de datos que mueve el grupo de datos de la orden

del almacenamiento temporal en el proceso. (El proceso fuera de línea, si se debe medir, implica otro proceso funcional, independiente. Efectivamente, el usuario funcional ha iniciado dos entradas desencadenantes, una inicia el proceso fuera de línea para cargar las órdenes al almacenamiento temporal y el otro para comenzar el procesamiento por lotes de las órdenes.)

APLICACIÓN DE NEGOCIO 6: Supóngase un FUR para un proceso por lotes a final de año para informar de los resultados del negocio durante el año y para reajustar puestos para comienzos del año que viene. Físicamente, la marca de fin de año (tick) de un reloj originado por el sistema operativo ocasiona que la aplicación comience el proceso. Lógicamente, sin embargo, cada proceso funcional de la aplicación toma sus datos de entrada del flujo de datos a ser procesado por lotes. Esto debe ser analizado de modo normal (por ejemplo los datos de entrada para cualquier proceso funcional constan de una o más Entradas, la primera de las cuales es la Entrada desencadenante de dicho proceso).

Sin embargo, asúmase que hay un proceso funcional particular en la aplicación que no requiere ningún dato de entrada para producir sus series de informes. Físicamente, el usuario funcional (humano) ha delegado en el sistema operativo la tarea de desencadenar este proceso funcional. Puesto que todos los procesos funcionales deben tener una Entrada desencadenante, podríamos considerar que la marca de tiempo (tick) de fin de año del reloj ha iniciado el flujo de procesamiento por lotes, desempeñando este papel para este proceso. Este proceso funcional podría entonces necesitar diferentes Lecturas y muchas Salidas para producir sus informes.

Lógicamente, el análisis de este ejemplo no es diferente si el usuario funcional humano inicia la producción de uno o más informes vía un clic del ratón en un icono del menú on-line, en vez de delegar la producción del informe al sistema operativo vía flujo por lotes.

Para varios ejemplos sobre distinguir los eventos desencadenantes y procesos funcionales en flujos de proceso por lotes, consulte la 'Guía para medición de aplicaciones de negocio utilizando COSMIC' [7], sección 4.6.3.

3.2.4 Eventos disparadores y procesos funcionales en el ámbito de aplicaciones de tiempo real

- a) Un evento desencadenante es detectado típicamente por un sensor.

TIEMPO REAL EJEMPLO 1: Cuando un sensor (usuario funcional) detecta que la temperatura alcanza un valor determinado (evento desencadenante), se produce que el sensor (usuario funcional) envíe una señal iniciando un movimiento de datos de Entrada desencadenante de un proceso funcional para apagar un calefactor (proceso funcional).

TIEMPO REAL EJEMPLO 2: Un avión militar tiene un sensor que detecta el evento 'misil acercándose'. El sensor es un usuario funcional del software embebido que debe responder a la amenaza. Para este software, solo cuando el sensor detecta algo ocurre un evento, y es el sensor (el usuario funcional) el que pone en funcionamiento al software generando un grupo de datos para iniciar una Entrada desencadenante enviando un mensaje diciendo por ejemplo: 'el sensor 2 ha detectado un misil', además quizás también envía un flujo de datos sobre la velocidad del misil que se acerca y sus coordenadas. El misil es el objeto de interés

- b) Señales periódicas de una reloj (marca de tiempo (ticks) de reloj) pueden desencadenar un proceso funcional

Ver Ejemplo 2 en tiempo real de la sección 3.2.1. No hay otros datos que acompañan a la marca de tiempo (tick). El proceso funcional obtiene datos de varios sensores y toma cualquier acción que se necesita.

3.2.5 Más sobre procesos funcionales separados

El software diferencia eventos y proporciona los correspondientes procesos funcionales dependiendo solo de sus FUR. Al medir el software puede a veces ser difícil decidir qué eventos separados se requiere que el software reconozca. Este es especialmente el caso cuando los FUR originales no están ya disponibles y cuando, por ejemplo, los que desarrollan el software haya visto que es económico combinar varios requisitos. Podría ayudar con el análisis examinar la organización de los datos de entrada (ver abajo) o examinar los menús de algún software instalado para ayudar a distinguir los eventos separados a los que el software debe responder y los correspondientes procesos funcionales

APLICACIÓN DE NEGOCIOS EJEMPLO 1: Suponga que hay un requisito de un usuario funcional para dos tipos de beneficios sociales, el primero para un niño adicional y segundo para un 'crédito fiscal de trabajo' para los que tienen un sueldo bajo, estos son requisitos a los que el software debe responder como dos eventos separados en el mundo de los usuarios funcionales humanos. Por tanto, debería haber dos procesos funcionales, aunque solo se haya utilizado un formulario de impuestos para recoger los datos en ambos casos.

De acuerdo con el inciso c) de su definición, un proceso funcional debe 'cumplir su FUR para todas las posibles respuestas a la entrada desencadenante'. Esto significa que el mismo *tipo* de proceso funcional debe ser capaz de hacer frente a todas las posibles *ocurrencias* de valores de los atributos de datos del grupo de datos movido por su entrada desencadenante, incluyendo tanto los valores de datos válidos y no válidos, e incluso en algunos casos valores de datos faltantes. Todas estas variaciones en los valores de los datos movidos por la entrada desencadenante por lo general tienen como resultado diferentes rutas de procesamiento que son seguidos cuando el proceso funcional se ejecuta. Pero a pesar de todas estas variaciones, todavía tenemos que identificar sólo el tipo de proceso funcional iniciado por su único Tipo de Entrada desencadenante. (El medidor sólo necesita identificar todos los movimientos de datos de este proceso funcional; las diversas rutas de procesamiento en las que pueden ocurrir son irrelevantes para la medición.)

APLICACIÓN DE NEGOCIOS EJEMPLO 2: Un proceso funcional que ofrece una capacidad de búsqueda general contra una base de datos puede ser obligado a aceptar hasta cuatro parámetros de búsqueda (atributos de la Entrada desencadenante). Pero el mismo proceso funcional funcionará si se introducen los valores de sólo uno, dos o tres parámetros de búsqueda.

APLICACIÓN DE NEGOCIOS EJEMPLO 3: Para un proceso funcional que registra un nuevo cliente de una empresa de alquiler de coches, es obligatorio introducir los datos para la mayoría de los atributos de datos, pero algunos (por ejemplo, algunos datos de contacto) son opcionales y puede dejarse en blanco. Independientemente de si se introducen todos o un subconjunto de estos atributos sólo hay un proceso funcional para registrar un nuevo cliente.

APLICACIÓN DE NEGOCIOS EJEMPLO 4: Continuando con el ejemplo 3, para el proceso funcional para hacer una reserva de alquiler de coches en la misma empresa, hay varias opciones que pueden o no pueden tomarse, por ejemplo, para el seguro extra, conductores adicionales, las peticiones de los asientos para niños, etc. Estas diferentes opciones conducen a diferentes rutas de procesamiento dentro del proceso funcional alquilar un coche, pero todavía hay un solo proceso funcional para reservar un alquiler de coches.

TIEMPO REAL EJEMPLO: Una Entrada desencadenante (información altitud de la aeronave enviado por el Sistema de Posicionamiento Geográfico) a un proceso funcional de un sistema de aeronáutica desencadenará una de las dos rutas de procesamiento muy diferentes dentro del proceso funcional dependiendo del valor de la entrada, es decir, si la altitud es superior o por debajo de una altura dada. Las diferentes rutas mostrarán distintos grupos de datos en el mapa del piloto y, si la altura es demasiado bajo, se emitirán advertencias adicionales. Sólo hay un proceso funcional.

Una vez identificados, cada proceso funcional puede ser registrada en una línea individual, bajo la capa apropiada y la pieza de software, en la matriz de Modelo Genérico de Software (Apéndice A).

3.2.6 La medición de los componentes de un sistema de software distribuido

Cuando el propósito de una medición es medir por separado el tamaño de cada componente de un sistema de software distribuido, un alcance de la medición independiente debe definirse para cada componente. En tal caso, el dimensionamiento de los procesos funcionales de cada componente sigue todas las reglas normales como ya se ha descrito.

Desde el proceso para cada medición (... definir el alcance, los usuarios funcionales y la frontera, etc ...) se deduce que si una pieza de software consiste en dos o más componentes, no puede haber ninguna superposición entre el alcance de la medición de cada componente. El alcance de la medición para cada componente debe definir un conjunto de procesos funcionales completos. Por ejemplo, no puede haber un proceso funcional que esté dentro de dos alcances definidos, parte en un alcance definido y parte en otro. Asimismo, los procesos funcionales dentro del alcance de medición para uno de los componentes no tienen ninguna información sobre los procesos funcionales dentro del alcance de otro componente, a pesar de que los dos componentes intercambien mensajes.

El usuario funcional (o usuarios funcionales) de cada componente se determina examinando donde ocurren los eventos que hacen que se desencadenen los procesos funcionales en el componente que está siendo examinado. (Eventos desencadenantes sólo pueden ocurrir en el mundo de un usuario funcional).

La Figura 3.7 ilustra los procesos funcionales de los dos componentes de un sistema distribuido de cliente-servidor y los movimientos de datos que se intercambian.

3.2.7 Independencia de los procesos funcionales compartiendo algunas funciones comunes o similares: Reutilización

Cualquiera de dos o más procesos funcionales en el mismo software que se está midiendo puede tener alguna funcionalidad que es idéntica o muy similar en cada proceso. Este fenómeno se conoce como 'funcionalidad coincidente', o 'similitud funcional'.

Sin embargo, en el método COSMIC (como en todos los demás métodos FSM) cada proceso funcional se define, modelado y medido independientemente, es decir, sin hacer referencia a cualquier otro proceso funcional en el mismo software que se está midiendo (ver el inciso a de la definición de un proceso funcional). Cualquier funcionalidad requerida que es común o similares a través de dos o más procesos funcionales en el mismo software que se está midiendo debe tenerse en cuenta en cada uno de estos procesos funcionales. Los siguientes son ejemplos de coincidencia o similitud funcional que se puede encontrar en la práctica.

APLICACIÓN DE NEGOCIOS EJEMPLO: Varios procesos funcionales en el mismo software que se miden pueden necesitar la misma funcionalidad de validación de ejemplo 'fecha de la orden', o puede tener que acceder a los mismos datos persistentes, o puede que tenga que realizar el mismo cálculo de intereses.

TIEMPO REAL EJEMPLO: Varios procesos funcionales en el mismo software que se mide puede ser necesario obtener datos del mismo sensor (movimiento común de un mismo grupo de datos) o pueden necesitar la misma escala para llevar a cabo el cálculo de una conversión, por ejemplo, de Fahrenheit a Centígrados (manipulación de datos común)

Cuando una especificación de FUR se implementa en software, cualquier 'funcionalidad coincidente' puede o no puede ser desarrollado como reutilizable software. Todas las decisiones de implementación, incluyendo la extensión de software de re-uso real o potencial deben ser ignoradas en la medición de tamaño funcional. Sin embargo, puede necesitarse tomar en cuenta la reutilización cuando las mediciones de tamaño funcional se utiliza para el análisis de esfuerzo del proyecto o propósitos de estimación.

3.2.8 Eventos que desencadenan la ejecución de un sistema de software

Al medir el tamaño de una pieza de software, se identifican sólo los eventos y las entradas correspondientes desencadenante que desencadenan los procesos funcionales a los que el software debe responder a como se define en su FUR.

Funcionalidad necesaria para la puesta en marcha (o 'lanzamiento') del software en sí mismo no es parte de estos procesos funcionales y deben ser ignorados (o medirse por separado, si es necesario). Los siguientes ejemplos describen cómo se inicia el software en tres ámbitos.

APLICACIÓN DE NEGOCIOS EJEMPLO: Para una aplicación de negocio, el usuario funcional que inicia la aplicación puede ser un componente planificador del sistema operativo, un operador de computadora, o cualquier otro usuario humano (por ejemplo, cuando un usuario de PC lanza un software de navegación o de procesador de palabras).

TIEMPO REAL EJEMPLO: Para una aplicación en tiempo real, el usuario funcional que inicia la aplicación puede ser el sistema operativo o el administrador de la red generando una señal de reloj, o un operador humano (por ejemplo, para iniciar un sistema de control de proceso desde una estación de trabajo del operador).

INFRAESTRUCTURA EJEMPLO: Para un sistema operativo de la computadora, el usuario funcional que inicia el sistema operativo es un programa de arranque que se inicia cuando la alimentación del ordenador está encendido.

Los siguientes son ejemplos en dos ámbitos distintos de las relaciones que pueden existir, en su caso, entre el evento/proceso que se inicia el software a ser medido y los eventos/procesos que el software debe ejecutar como se describe en los FUR.

APLICACIÓN DE NEGOCIOS EJEMPLO 1: Una aplicación para procesar los datos de entrada para una variedad de procesos funcionales mediante procesamiento por lotes, puede ser iniciado por un programa de calendarización del sistema operativo. Si el propósito es medir los FUR de la aplicación de procesamiento por lote, la funcionalidad de 'inicializar el sistema' debe ser ignorada. Las entradas desencadenantes para los procesos funcionales de la aplicación de procesamiento por lote y cualesquiera otras Entradas que puedan ser necesarias formarán los datos de entrada para la aplicación de procesamiento por lote.

APLICACIÓN DE NEGOCIOS EJEMPLO 2: Excepcionalmente, una aplicación de procesamiento por lotes para producir informes de resumen al final de un período de tiempo se puede iniciar sin necesidad de ningún dato de entrada proporcionado directamente por el usuario funcional. Para el análisis ver Ejemplo 6 de aplicaciones de negocio en la sección 3.2.3.

TIEMPO REAL EJEMPLO: Un vehículo moderno tiene un sistema distribuido de Unidades de Control Electrónico (ECU) para controlar muchas funciones, por ejemplo, de gestión del motor, frenos, aire acondicionado, etc. En la arquitectura AUTOSAR, en un sistema distribuido, el módulo "Gestión de Red" (NM), que siempre se está ejecutando, es responsable de la activación de ECU que están conectados entre sí a través de una red ('bus'). Este módulo también se ocupa de la conmutación coordinada entre los estados de operación de la ECU: funcionamiento normal, de baja potencia y de reposo. Por lo tanto, el NM el que se despierta o pone en reposo al ECU. Al medir cualquier software de aplicación ECU, esta funcionalidad NM debe ser ignorada.

3.3 Identificación objetos de interés y grupos de datos

3.3.1 Definiciones y principios

Una vez identificados los procesos funcionales, el siguiente objetivo principal debe ser la identificación de sus movimientos de datos. Para ello debemos recordar el Modelo Genérico de Software (ver sección 1.3.2) y la introducción de los conceptos de este capítulo en la sección 3.0, que establecen que un movimiento de datos 'mueve un grupo de datos, del cual todos sus atributos de datos describen un solo 'objeto de interés'. Por lo tanto, para entender un movimiento de datos, primero tenemos que definir y comprender estos tres conceptos.

DEFINICIÓN – Objeto de interés

Cualquier 'cosa' en el mundo del usuario funcional que se identifica en los Requisitos Funcionales de Usuario, sobre la que se requiere que el software procese y/o almacene datos. Puede ser cualquier cosa física, así como cualquier objeto conceptual o parte de un objeto conceptual.

NOTA 1: En la definición anterior, 'procesar' puede significar cualquier operación para mover y/o manipular los datos.

NOTA 2: En el método COSMIC, el término 'objeto de interés' se utiliza para evitar los términos relacionados con métodos de ingeniería de software específicos. El término no implica 'objetos' en el sentido utilizado en métodos orientados a objetos.

DEFINICIÓN – Grupo de datos

Un grupo de datos es un conjunto único, no vacío, no ordenado y no redundante de atributos de datos donde cada atributo de datos incluido describe un aspecto complementario del mismo objeto de interés.

Una vez identificado, cada grupo de datos candidato debe cumplir con el siguiente principio:

PRINCIPIO – Grupo de datos
Cada grupo de datos identificado deberá ser único y distinguible a través de su colección única de atributos de datos.

Una vez identificado, cada grupo de datos puede estar registrado en una columna individual en la matriz del Modelo Genérico de Software (Apéndice A), bajo la correspondiente etiqueta.

3.3.2 Sobre la materialización de un grupo de datos

En la práctica, la materialización de un grupo de datos puede tomar muchas formas, por ejemplo:

- a) Como una estructura física grabada en un dispositivo de almacenamiento continuo (archivo, tabla de base de datos, memoria ROM, etc.)
- b) Como una estructura física en la memoria volátil del ordenador (estructura de datos asignada dinámicamente o a través un bloque pre-asignado de espacio de memoria)
- c) Como la presentación agrupada de atributos de datos relacionados funcionalmente en un dispositivo de entrada/salida (pantalla de visualización, informe impreso, panel de control de visualización, etc.)
- d) Como un mensaje transmitido entre un dispositivo y un ordenador, o sobre una red, etc.

3.3.3 Sobre la identificación de objetos de interés y grupos de datos

Las definiciones y los principios de los objetos de interés y de los grupos de datos son intencionalmente amplios con el objetivo de que se puedan aplicar al rango de software más amplio posible. Esta cualidad hace que algunas veces resulte difícil de aplicar la definición y los principios cuando se mide una parte específica del software. Por tanto, los casos siguientes y los ejemplos se han diseñado para ayudar en la aplicación de los principios a casos específicos.

Al enfrentarnos a la necesidad de analizar un grupo de atributos de datos que son movidos dentro o fuera de un proceso funcional o que son movidos por un proceso funcional a o desde un almacén persistente, *es de crítica importancia* decidir si todos los atributos comunican datos sobre un único 'objeto de interés', ya que es éste último el que determina el número de 'grupos de datos' separados tal y como los define el método COSMIC. Por ejemplo, si los atributos de datos que van a ser los datos de entrada de un proceso funcional son atributos de tres objetos de interés separados, entonces necesitamos identificar tres movimientos de datos de Entrada separados.

Objetos de interés y grupos de datos en el ámbito de las aplicaciones de negocio

APLICACIÓN DE NEGOCIOS EJEMPLO 1: En el ámbito de software de aplicaciones de negocio, un objeto de interés podría ser 'empleado' (físico) o 'pedido' (conceptual), asumiendo que se requiere que el software almacene datos sobre empleados o pedidos. En el caso de pedidos, normalmente desde el FUR de pedidos multilínea se identifican dos objetos de interés: 'pedido' y 'detalle de pedido'. Los grupos de datos correspondientes podrían llamarse 'datos de pedidos' o 'detalle de datos de pedidos'.

Los grupos de datos se forman cuando sea que haya una consulta ad hoc que pida datos sobre una 'cosa', es decir un 'objeto de interés', que no se tiene en el almacén persistente, pero que se puede derivar de datos que están en el almacén persistente. El movimiento de datos de Entrada en una consulta ad hoc (los parámetros de selección para derivar los datos requeridos) y el movimiento de datos de Salida (que contienen los atributos deseados), ambos mueven grupos de datos sobre dicha "cosa". Estos son grupos de datos transitorios que no sobreviven a la ejecución del proceso funcional. Son grupos de datos válidos porque cruzan el límite o frontera entre el software y su usuario o usuarios.

APLICACIÓN DE NEGOCIOS EJEMPLO 2: Dada una consulta ad hoc contra una base de datos de personal para extraer una lista de nombres de todos los empleados de más de 35 años. La Entrada mueve un grupo de datos que contiene los parámetros de la selección. La Salida mueve un grupo de datos que contiene solo el atributo 'nombre'; el objeto de interés (o 'cosa') de los grupos de datos movidos por la Entrada y la Salida es 'todos los empleados de más de 35 años'. Es importante nombrar claramente un grupo de datos transitorio en relación con su objeto de interés, por ejemplo, 'Empleados de más de 35'. No relacionar el nombre del grupo de datos transitorio al objeto (s) de interés a partir del cual el grupo de datos transitorio se deriva (por ejemplo, el 'nombre del empleado').

Para una discusión detallada sobre métodos de análisis de datos para determinar objetos de interés y grupos de datos separados, el lector puede ir a 'Guía para Medir Aplicaciones de Negocio utilizando COSMIC' [7].

Objetos de interés y grupos de datos en el ámbito de las aplicaciones en tiempo real

TIEMPO REAL EJEMPLO 1: Movimientos de datos que son Entradas desde dispositivos físicos típicamente contienen datos sobre el estado de un solo objeto de interés, tales como si un válvula está abierta o cerrada, o indican un tiempo en el que los datos a corto plazo, el almacenamiento volátil es válido o inválido, o contienen datos que indican que ha ocurrido un evento crítico y qué causa una interrupción. De un modo similar un comando de Salida para encender o apagar una lámpara de aviso comunica datos sobre un solo objeto de interés.

TIEMPO REAL EJEMPLO 2: Un software de intercambio de mensajes puede recibir un grupo de datos de mensajes tales como una Entrada y guiarlo adelante sin cambios como una Salida, como FUR de una parte del software en particular. Los atributos del grupo de datos de mensajes podrían ser, por ejemplo, "id del mensaje, el que lo envía, el que recibe, ruta-código y mensaje-contenido; el objeto de interés del mensaje es 'mensaje'.

TIEMPO REAL EJEMPLO 3: Una estructura de datos habitual, representando objetos de interés que se mencionan en los FUR, que puede ser mantenida por procesos funcionales, y que es accesible para la mayoría de procesos funcionales encontrados en el software medido.

TIEMPO REAL EJEMPLO 4: Una estructura de datos de referencia, representando gráficos o tablas de valores encontrados en los FUR, los cuales están en memoria permanente (memoria ROM por ejemplo) y accesible para la mayoría de procesos funcionales encontrados en el software medido.

TIEMPO REAL EJEMPLO 5: Archivos, comúnmente designados como 'archivos planos', representando objetos de interés mencionados en los FUR, los cuales se mantienen en un dispositivo de almacenamiento persistente.

3.3.4 Grupos de datos o datos que NO son candidatos a movimientos de datos

Datos cualesquiera que aparezca en pantallas de entrada o salida o en informes y que no están relacionados con un objeto de interés para un usuario funcional NO deberían ser identificados indicando un movimiento de datos, por tanto no deberían medirse.

APLICACIÓN DE NEGOCIOS EJEMPLO 1: Datos generales de la aplicación tales como encabezados y pies de página (nombre de la compañía, nombre de la aplicación, fecha del sistema, etc.) que aparecen en todas las pantallas.

APLICACIÓN DE NEGOCIOS EJEMPLO 2: 'Comandos de control' (un concepto definido solo en el ámbito de aplicaciones de gestión) que permite a un usuario funcional controlar el uso del software más que mover datos, por ejemplo los comandos de página arriba/abajo, pulsar "OK" para admitir un mensaje de error, etc. - ver la sección 3.5.10.

El Modelo Genérico de Software COSMIC asume que toda manipulación de datos dentro de un proceso funcional está asociada con unos de los cuatro tipos de movimientos de datos- ver sección 3.5.6. Por tanto ningún movimiento o manipulación de datos dentro de un proceso funcional puede ser identificado como candidato para movimiento de datos junto con las Entradas, Salidas, Lecturas y Escrituras. (Para ver ejemplos de manipulación y movimientos de datos que puedan ser mal interpretados como movimientos de datos ver sección 3.5.4, principio c) para Lecturas, y sección 3.5.5 principio d) para Escrituras.

3.3.5 El usuario funcional como objeto de interés

En muchos casos simples de software en tiempo real como el descrito en el Ejemplo 1 de la sección 3.3.3 de arriba, la Fase de Estrategia puede mostrar que un dispositivo físico tal como un sensor podría ser un usuario funcional. Más tarde, en la Fase de Representación, este mismo sensor puede también ser identificado como un objeto de interés. Esto es simplemente porque el dispositivo físico 'interactúa con el software' y, al mismo tiempo que el dispositivo físico es una "cosa" sobre el que se requiere el software para procese y/o almacene datos'. En efecto, el dispositivo físico interactúa con el software a través de la frontera para proporcionar o para recibir información sobre sí mismo en forma de un grupo de datos que describe algún aspecto de sí mismo.

TIEMPO REAL EJEMPLO: Supóngase que un sensor de temperatura 'A' envía una medida de la temperatura actual de un material para su procesamiento por un proceso funcional. Desde un punto de vista el sensor puede ser considerado como el suministro de información sobre su propio estado. Por tanto, el sensor cumple los criterios de identificación como un objeto de interés y puede ser identificado desde los FUR como tal. Sin embargo, el sensor físico está en el mundo de los usuarios y '... interactúa con el software ... a través de la frontera', por lo tanto, también cumple con los criterios para la identificación como usuario funcional y otra vez puede ser asignado como tal. Así que el sensor de temperatura se podría añadir a la Modelo Contextual de Software, y el objeto de interés 'sensor de temperatura' podría añadirse al Modelo Genérico de Software.

3.4 Identificación de los atributos de datos (opcional)

Esta sección discute la identificación de atributos de datos referenciados por la pieza del software que se va a medir. En el método COSMIC no es obligatorio identificar los atributos de los datos. Sin embargo, la comprensión del concepto de 'atributo de datos' es necesario para comprender la sección sobre 'cambios de medición ...', donde un FUR que cambia un atributo de datos puede resultar en un movimiento de datos debiendo ser medido. También, puede ser útil para analizar e identificar los atributos de datos en el proceso de distinguir grupos de datos y objetos de interés, y los atributos de datos también puede ser identificada si se requiere una subunidad de la medida de tamaño, tal y como se presenta en la sección 4.5 'Extendiendo el método de medición COSMIC'

3.4.1 Definición

DEFINICIÓN – Atributo de datos

La pieza más pequeña de información, dentro de un grupo de datos identificados, que tiene un significado desde la perspectiva de los requisitos funcionales del software.

APLICACIÓN DE NEGOCIOS EJEMPLO: Atributos de datos en el contexto del ámbito de aplicaciones de negocio son por ejemplo elementos de datos registrados en el diccionario de datos y atributos de datos que aparecen en un modelo de datos conceptual o lógico.

REAL-TIME EXAMPLE: Atributos de datos en el contexto de las aplicaciones de software de tiempo real son por ejemplo 'señal', representando una señal recibida desde un sensor y 'De (dirección), A (Dirección), Contenido', representando partes de un mensaje en transmisión.

3.4.2 Sobre la asociación de atributos de datos y grupos de datos

En teoría, un grupo de datos puede contener solo un atributo de datos si esto es suficiente, desde la perspectiva de los requisitos funcionales de usuario, para describir el objeto de interés. En la práctica, dichos casos ocurren normalmente en las aplicaciones de software en tiempo real (por ejemplo: la Entrada que transmite la marca de tiempo (el tick) de un reloj a tiempo real o la lectura de un sensor); éstos son menos comunes en el software de negocio.

3.5 Identificando los movimientos de datos

Este paso consiste en identificar los movimientos de datos (Entrada, Salida, Lectura y Escritura) de cada proceso funcional.

3.5.1 Definición de los tipos de movimientos de datos

DEFINICIÓN – Movimiento de datos

Un componente funcional base que mueve un único tipo de grupo de datos.

NOTA 1: Hay cuatro subtipos de movimiento de datos (-tipo) llamados: Entrada, Salida, Lectura y Escritura (-tipos)

NOTA 2: Para propósitos de medición, cada tipo de movimiento de datos se considera que incluye ciertas manipulaciones de datos asociadas - Ver sección 3.5.6 para más detalles.

NOTA 3: Más precisamente, es una *ocurrencia* de un movimiento de datos, no un tipo de movimiento de datos, que realmente mueve las *ocurrencias* del grupo de datos (no los tipos). Este comentario se aplica también a las definiciones de Entrada, Salida, Lectura y Escritura.

DEFINICIÓN – Entrada, (Entry, E)

Un movimiento de datos que mueve un grupo de datos desde un usuario funcional a través de la frontera hacia el proceso funcional donde se necesita.

NOTA: Se considera que una Entrada incluye ciertas manipulaciones de datos asociadas (ver sección 3.5.6)

DEFINICIÓN – Salida, (Exit, X)

Un movimiento de datos que mueve un grupo de datos desde un proceso funcional a través de la frontera hacia el usuario funcional que los requiere.

Nota: Se considera que una Salida incluye ciertas manipulaciones de datos asociadas (ver sección 3.5.6)

DEFINICIÓN – Lectura (Read, R)

Un movimiento de datos que mueve un grupo de datos desde un almacén persistente en el proceso funcional que los requiere.

Nota: Se considera que una Lectura incluye ciertas manipulaciones de datos asociadas (ver sección 3.5.6)

DEFINICIÓN – Escritura (Write, W)

Un movimiento de datos que mueve un grupo de datos a un almacén persistente que se encuentra dentro de un proceso funcional.

Nota: Se considera que una Escritura incluye ciertas manipulaciones de datos asociadas (ver sección 3.5.6)

La figura 3.5, debajo, ilustra la relación general existente entre los cuatro tipos de movimientos de datos, el proceso funcional al que pertenecen y la frontera del software medido

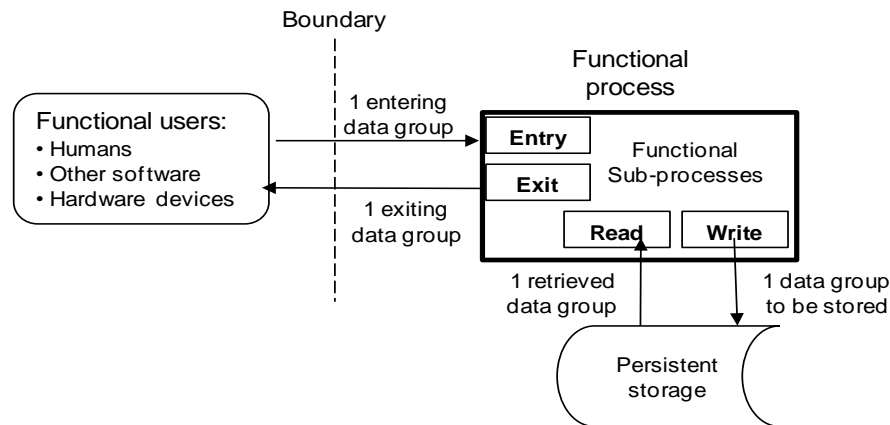


Figura 3.5 - Los cuatro tipos de movimiento de datos, cada uno moviendo un grupo de datos, y su relación con un proceso funcional. (Un proceso funcional puede, por supuesto, tener, muchos movimientos de datos E, X, R, y W)

3.5.2 Identificando entradas (E)

Un candidato a movimiento de datos de Entrada debe cumplir con los siguientes principios:

PRINCIPIOS – Entradas (E)	
a)	Una Entrada deberá mover un único grupo de datos describiendo un solo objeto de interés de un usuario funcional a través de la frontera y hacia un proceso funcional del que la entrada forma parte. Si la entrada a un proceso funcional comprende más de un grupo de datos, cada uno describiendo un objeto de interés diferente, se identifica una entrada por cada grupo de datos que entra. (Véase también la sección 3.5.7 sobre ‘Movimientos de datos únicos y posibles excepciones’)
b)	Una Entrada no deberá sacar datos a través de la frontera, ni leer ni escribir datos del/hacia el almacenamiento persistente.

Las siguientes reglas ayudan a confirmar la condición de un candidato a movimiento de datos de Entrada:

REGLAS – Entradas (E)	
a)	El grupo de datos de una Entrada desencadenante puede constar de sólo un atributo de datos que simplemente informa al software que ‘un evento “Y” ha ocurrido’. Muy a menudo, especialmente en el software de aplicaciones de negocio, el grupo de datos de una Entrada desencadenante tiene varios atributos que informan al software que ‘un evento “Y” ha ocurrido y aquí están los datos sobre ese evento en particular’.
b)	Las marcas de tiempo (ticks) de reloj que están desencadenando eventos serán siempre externos al software que está siendo medido. Por lo tanto, por ejemplo, una marca de tiempo (tick) de reloj cada 3 segundos estará asociada a una Entrada que mueve un grupo de datos con un sólo atributo. Se debe notar que no hay ninguna diferencia si el evento desencadenante es generado periódicamente por el hardware o por otra parte del software fuera de los límites del software medido.

- c) Salvo que sea necesario en un proceso funcional específico, obtener el tiempo desde el reloj del sistema, este no será considerado como causa de una Entrada.
- d) Si una ocurrencia de un evento específico desencadena la Entrada de un grupo de datos que comprende hasta “n” atributos de un objeto de interés en particular y los FUR permiten que otras ocurrencias del mismo evento puedan desencadenar una Entrada de un grupo de datos que tiene valores de atributos de sólo un subconjunto de ‘n’ atributos del objeto de interés, entonces será identificada una entrada, compuesto por todos ‘n’ atributos.
- e) Cuando se identifican las Entradas en una pantalla que permite a los usuarios funcionales humanos introducir los datos de entrada en los procesos funcionales, se debe analizar sólo las pantallas que están llenos de datos. Ignore cualquier pantalla con formato pero de otra manera ‘en blanco’ a excepción de posibles valores por defecto, y no hacer caso de todos los campos y otros encabezados que permiten a los usuarios humanos comprender los datos de entrada requeridos.

NOTA. Puede que sea necesario tener en cuenta los campos y otros encabezados en la medición de cambios de los FUR en las Entradas - ver sección 4.4.1.

APLICACIÓN DE NEGOCIOS EJEMPLO ilustra regla c): Cuando un proceso funcional añade una marca de tiempo a un registro que se hizo persistente o fue una salida, no se identifica una Entrada para la obtención del valor del reloj del sistema.

Una vez identificados, cada movimiento de datos de Entrada se puede registrar marcando la casilla correspondiente a la tabla del Modelo Genérico de Software (apéndice A) con una “E”.

3.5.3 Identificando de salidas (X)

Un candidato a movimiento de datos de Salida debe cumplir con los siguientes principios:

PRINCIPIOS – Salidas (X)

- a) Una Salida deberá mover un único grupo de datos describiendo un solo objeto de interés desde el proceso funcional del que la salida forma parte a través de la frontera hacia un usuario funcional. Si la salida de un proceso funcional comprende más de un grupo de datos, hay que identificar una salida para cada grupo de datos que sale. (Véase también la sección 3.5.7 sobre ‘Movimientos de datos únicos y posibles excepciones’.)
- b) Una Salida no introducirá datos a través de la frontera, ni leer ni escribir datos del/hacia el almacenamiento persistente.

Las siguientes reglas pueden ser útiles para confirmar el estado de un movimiento de datos de Salida candidato:

REGLAS – Salidas (X)

- a) Una consulta que emite el texto fijo, (donde ‘fijo’ significa que el mensaje no contiene valores de datos variables, por ejemplo, el resultado de pulsar un botón de ‘Términos y Condiciones’ en un sitio web comercial), se modela como que tiene una sola Salida por la salida de texto fijo.

NOTA: Para la salida de la funcionalidad de ‘Ayuda’, consulte la ‘Guía para el dimensionamiento de Aplicaciones de Software de Negocio’. Para la salida de los mensajes relacionados con las condiciones de error o confirmación de éxito, véase la sección 3.5.11 de este Manual de Medición.

- b) Si la Salida de un proceso funcional mueve un grupo de datos que comprende hasta 'n' atributos de datos de un objeto particular de interés y los FUR permiten que el proceso funcional pueda tener una ocurrencia de una Salida que mueve un grupo de datos que tiene valores para un sub-conjunto de solamente 'n' atributos del objeto de interés, entonces se identifica una Salida, que comprende todos los 'n' atributos de datos.
 - b) Cuando se hace la identificación de Salidas, ignorar todos los campos y encabezados que permiten a los usuarios humanos comprender los datos de salida.
- NOTA: Puede que sea necesario tener en cuenta el campo y otros encabezados en la medición de cambios de los FUR en Salidas - véase la sección 4.4.1.

Una vez identificados, cada movimiento de datos de Salida puede ser registrado marcando la casilla correspondiente de la tabla del Modelo Genérico de Software (apéndice A) con un "X".

Véase también el apartado 3.5.11 para saber la manera de identificar los movimientos de datos de Salida para mensajes de error.

3.5.4 *Identificando Lecturas (R)*

Un candidato a movimiento de datos de Lectura debe cumplir con los siguientes principios:

PRINCIPIOS – Lecturas (R)	
a)	Una Lectura deberá mover un único grupo de datos describiendo un solo objeto de interés del almacén persistente a un proceso funcional del cual la lectura forma parte. Si el proceso funcional debe recuperar más de un grupo de datos del almacén, se debe identificar una Lectura para cada grupo de datos que es recuperado. (Véase también la sección 3.5.7 sobre los 'Movimientos de datos únicos y posibles excepciones'.)
b)	Una Lectura no recibirá ni sacará datos a través de la frontera ni escribirá datos al almacenamiento persistente.
c)	Durante un proceso funcional, el movimiento o manipulación de constantes o variables que son internas del proceso funcional y que sólo se pueden cambiar por un programador, o mediante los resultados intermedios en un cálculo, o de los datos almacenados en un proceso funcional sólo resultantes de la aplicación, más que de los FUR, no se considerará como un movimiento de datos de Lectura.
d)	Una Lectura siempre incluye cualquier funcionalidad de 'solicitud de lectura' (así, un movimiento de datos separado nunca será contado para cualquier funcionalidad de 'solicitud de lectura'). Véase también la sección 3.5.9.

Las siguientes reglas pueden ser útiles para confirmar el candidato como movimiento de datos de Lectura:

REGLAS – Lecturas (R)	
a)	Identifica una Lectura cuando, según los FUR, el software que se está midiendo debe recuperar un grupo de datos desde el almacenamiento persistente.
b)	No identificar una Lectura cuando los FUR del software que se está midiendo especifican algún usuario funcional de software o hardware como la fuente de un grupo de datos, o como los medios de recuperación de un grupo de datos almacenados. (Para este caso ver los principios y reglas para las Entradas y Salidas.)

Una vez identificados, cada movimiento de datos de Lectura puede ser registrado marcando la casilla correspondiente de la tabla del Modelo Genérico de Software (apéndice A) con un “R”.

3.5.5 *Identificando escrituras (W)*

Un candidato como movimiento de datos de Escritura debe cumplir con los siguientes principios:

PRINCIPIOS – Escritura (W)
<p>a) Una Escritura deberá mover un único grupo de datos describiendo un solo objeto de interés del proceso funcional del que la Escritura forma parte hacia el almacén persistente. Si el proceso funcional debe pasar más de un grupo de datos al almacén persistente, identificar una Escritura por cada grupo de datos que se mueve al almacén persistente. (Véase también la sección 3.5.7 sobre ‘Movimientos de datos únicos y posibles excepciones’).</p> <p>b) Una escritura no recibirá ni sacará datos de la frontera, ni leerá los datos.</p> <p>c) Un requisito para borrar un grupo de datos de un almacén persistente se medirá como una sola Escritura.</p> <p>d) Lo siguiente no podrá considerarse como movimientos de datos de Escritura:</p> <ul style="list-style-type: none"> • El movimiento o manipulación de los datos que no existía en el inicio de un proceso funcional y que no se ha hecho persistente cuando el proceso funcional es completado; • Creación o actualización de variables o resultados intermedios que son internos al proceso funcional; • Almacenamiento de los datos por un proceso funcional resultante sólo de la aplicación, en lugar de que estén establecidos en los FUR. (Un ejemplo podría ser el almacenamiento de datos en forma temporal durante un largo proceso de ordenamiento en un trabajo de procesamiento por lotes.)

Las siguientes reglas pueden ser útiles para confirmar el estado de un movimiento de datos de Escritura candidato:

REGLAS – Escritura (W)
<p>a) Identificar una Escritura en que, de acuerdo a los FUR, el software que se está midiendo debe mover un grupo de datos hacia el almacenamiento persistente.</p> <p>b) No identificar una Escritura cuando los FUR del software que se está midiendo especifica algún usuario funcional de software o hardware como el destino del grupo de datos o como medio de almacenamiento del grupo de datos. (Para ver este caso los principios y reglas para las Entradas y Salidas.)</p>

Una vez identificados, cada movimiento de datos de Escritura puede ser registrado marcando la casilla correspondiente de la tabla del Modelo Genérico de Software (apéndice A) con un “W”.

3.5.6 *Sobre las manipulaciones de datos asociadas con los movimientos de datos*

Sub-procesos son, tal como se define en el principio (d) del Modelo Genérico de Software (ver sección 1.3), o bien movimientos de datos, o bien manipulaciones de datos. Sin embargo, por una convención de COSMIC (véase el principio (j) del Modelo Genérico de Software), no se reconoce la existencia separada de manipulación de datos y sub-procesos.

Toda manipulación de datos se considera que se explica por los movimientos de datos asociados. Por lo tanto la manipulación de datos puede ser ignorada EXCEPTO si hay un FUR que debe ser medido para un *cambio* a la manipulación de datos. Para estos casos es posible que se necesiten las reglas de esta sección para determinar qué tipo de manipulación de datos es asociada con qué tipo de movimiento de datos.

Una típica solicitud de cambio afecta tanto a los atributos movidos y la manipulación de datos asociada con un movimiento de datos en particular. Las reglas para medir los cambios necesarios de los FUR (ver sección 4.4 y la definición de una 'Modificación' en 4.4.1) son que si cualquier atributo (s) del grupo movido por un movimiento de datos o cualquiera de su manipulación de datos asociada debe ser cambiado, entonces el movimiento de datos debe ser identificado y medido como cambiado. Así que si un cambio requerido afecta sólo a la manipulación de los datos, necesitamos las siguientes reglas para identificar su movimiento de datos asociado (s) que se deben medir como cambiado.

DEFINICIÓN – Manipulación de datos

Todo lo que le sucede a los datos, distinto de un movimiento de datos en o de un proceso funcional, o entre un proceso funcional y almacén persistente.

El siguiente principio determina como el método COSMIC trata la manipulación de datos.

PRINCIPIO – Manipulación de datos asociada con movimientos de datos

Todas las manipulaciones de datos en un proceso funcional serán asociadas con los cuatro tipos de movimiento de datos (E, X, R, y W). Por convención, los movimientos de datos de un proceso funcional se supone que también representan la manipulación de los datos del proceso funcional.

REGLAS – Manipulación de datos asociada con movimientos de datos

- a) Un movimiento de datos de Entrada representa toda manipulación de datos para permitir que un grupo de datos sea introducido por un usuario funcional (por ejemplo, de formato y manipulaciones de presentación) y ser validado,
- b) Un movimiento de datos de Salida representa toda manipulación de datos para crear los atributos de datos de un grupo de datos que va a ser la salida y/o para permitir que el grupo de datos sea la salida (por ejemplo, de formato y manipulaciones de presentación) y para ser transferido al usuario funcional destinado,
- c) Un movimiento de datos de Lectura representa todo cálculo y/o procesamiento lógico necesario para recuperar un grupo de datos desde el almacenamiento persistente.
- d) Un movimiento de datos de Escritura representa todo cálculo y/o procesamiento lógico para crear o actualizar un grupo de datos que se escribe, o eliminar un grupo de datos.
- e) La manipulación de datos asociada con cualquiera de estos movimientos de datos no incluye ningún tipo de manipulación de datos que se necesita después de que el movimiento de datos se ha completado con éxito, ni tampoco incluye ningún tipo de manipulación de datos asociada con cualquier otro movimiento de datos.

APLICACIÓN DE NEGOCIOS EJEMPLO 1: Una Entrada incluye toda la manipulación necesaria para dar formato a una pantalla para permitir a un usuario humano introducir datos y validar los datos introducidos EXCEPTO cualquier Lectura(s) que pudiera requerirse para validar algún dato introducido o códigos, o para obtener algunas descripciones de los códigos asociados.

APLICACIÓN DE NEGOCIOS EJEMPLO 2: Una Salida incluye toda la manipulación para dar formato de salida y preparar algunos atributos de los datos para la impresión (o de salida en una pantalla), incluyendo los encabezados de campo legibles EXCEPTO cualquier Lectura o Entradas que podrían ser necesarias para proporcionar los valores o las descripciones de algunos de los atributos de los datos impresos.

3.5.7 Movimientos de datos únicos y posibles excepciones

El Modelo Genérico de Software supone que *normalmente* en cualquier proceso funcional *todos* los datos describen un objeto de interés que se requiere por ese proceso funcional, se introduce en un movimiento de datos de tipo Entrada y/o se lee en un movimiento de datos de tipo Lectura y/o se escribe en un movimiento de datos de tipo Escritura y/o saca información en un movimiento de datos de tipo Salida. El modelo asume, además, que toda manipulación de datos resultante de todos los valores posibles de los atributos de datos de un grupo de datos que se mueve está asociado con el movimiento de datos. También, dos movimientos de datos no pueden ser considerados como diferentes a menos que tengan diferente FUR para su manipulación de datos asociada, como *'toda'* manipulación de datos en un proceso funcional deberá ser asociada con los cuatro tipos de movimiento de datos (E, X, R y W) (ver el principio establecido en el apartado 3.5.6).

EJEMPLO que ilustra este último principio: Considere dos ocurrencias de un proceso funcional dado (- tipo). Supongamos que en la primera ocurrencia los valores de algunos atributos para ser movidos requieren un sub-proceso de manipulación de datos (- tipo) 'A' y que en otra ocurrencia del mismo proceso funcional los valores de atributo conducen a una manipulación de datos diferente sub-proceso (- tipo) 'B'. En tales circunstancias, ambas manipulaciones de datos de los subprocesos 'A' y 'B' deben estar asociados con el mismo movimiento de datos y por lo tanto sólo un movimiento de datos debe ser identificado y contabilizado en ese proceso funcional.

Puede, sin embargo, haber circunstancias en las que *distintos* grupos de datos que describen el *mismo* objeto de interés pueden ser requeridos (en el FUR) para moverse en un movimiento de datos del mismo tipo (E, R, W, X) en la misma funcional proceso.

Las siguientes reglas cubren la situación normal (regla a)) y otros posibles casos válidos (reglas b) y c)). La regla d) se refiere a las *ocurrencias* (en contraposición a los tipos) de movimientos de datos.

REGLAS – Movimientos de Datos Únicos y posibles excepciones

Nótese que todas las reglas de COSMIC que se refieren a tipos de usuarios funcionales, grupos de datos, movimientos de datos, procesos funcionales y objetos de interés. Para facilitar la lectura, normalmente omitimos 'tipo' de estos términos. Pero En la regla d) a continuación, incluimos 'tipo' donde es útil distinguir un 'tipo' de un 'acontecimiento'.

- a) A menos que los Requisitos Funcionales de Usuario sean dados como en las reglas c), todos los datos que describen cualquier objeto de interés que se requiere para ser introducido en un proceso funcional deberá ser identificado como un grupo de datos movido por una Entrada.

NOTA: Un proceso funcional puede, por supuesto, tener varias entradas, cada una moviendo datos que describen un objeto de interés diferente.

La misma regla equivalente se aplica a cualquier movimiento de datos de Lectura, Escritura o de Salida en cualquier proceso funcional.

- b) Un Requisito Funcional de Usuario puede especificar diferentes grupos de datos que deben introducirse en un proceso funcional por parte de los usuarios funcionales que deben ser identificados por separado para ese proceso funcional, donde cada grupo de datos describe el mismo objeto de interés. Una entrada se identificará para cada uno de estos diferentes grupos de datos.

La misma regla equivalente se aplica para las Salidas de datos a diferentes usuarios funcionales de cualquier proceso funcional.

NOTA: Cualquier proceso funcional debe tener sólo una Entrada desencadenante.

- c) Un Requisito Funcional de Usuario puede especificar diferentes grupos de datos a ser movidos desde el almacenamiento persistente a un proceso funcional, cada uno describiendo el mismo objeto de interés. Una Lectura se identificará para cada uno de estos diferentes grupos de datos. La misma regla equivalente se aplica para el Escrituras en cualquier proceso funcional dado.
- d) No se contarán las ocurrencias repetidas de cualquier tipo de movimiento de datos cuando se está ejecutando. El número de ocurrencias cuando se ejecuta software son irrelevantes para la medida del tamaño funcional.
- e) Esto se aplica incluso si múltiples ocurrencias del tipo de movimiento de datos difieren en su ejecución debido a diferentes valores de los atributos de datos del grupo de datos movido resultando en diferentes rutas de procesamiento siendo seguido por el tipo de proceso funcional.

Los siguientes ejemplos ilustran las reglas anteriores.

APLICACIÓN DE NEGOCIOS EJEMPLO 1 para la regla a): El caso normal es que una Escritura identificada mueva un grupo de datos que contiene todos los atributos de datos de un objeto de interés requerido para ser hecho persistente en un proceso funcional dado.

APLICACIÓN DE NEGOCIOS EJEMPLO 2 para la regla c) en contraste con la regla a): Supongamos un FUR para un solo proceso funcional A para almacenar dos grupos de datos derivados de los archivos de la cuenta corriente de un banco para su uso posterior por programas separados. El primer grupo de datos es el detalle de 'la cuenta en descubierto' (que incluye el atributo de saldo negativo). El segundo grupo de datos es el detalle 'de la cuenta de alto valor' (que sólo tiene el nombre y la dirección del titular de la cuenta, destinado a un marketing a través de correo). El proceso funcional A tendrá dos Escrituras, una para cada grupo de datos, ambos describiendo el mismo objeto de interés 'cuenta'.

TIEMPO REAL EJEMPLO 3 para la regla b): Se necesita un proceso funcional para aceptar diferentes grupos de datos a partir de dos sismógrafos diferentes (usuarios funcionales) que describen cada uno el mismo objeto de interés (el evento), por ejemplo, una explosión de ensayo. Identificar dos Entradas.

APLICACIÓN DE NEGOCIOS EJEMPLO 4 para la regla b): Supongamos que un FUR existe para un solo proceso funcional para producir dos o más Salidas moviendo diferentes grupos de datos que describen el mismo objeto de interés, destinadas a diferentes usuarios funcionales. Por ejemplo, cuando un nuevo empleado se une a una empresa un informe se produce al pasar el trabajador a firmar y validar sus datos personales, y un mensaje se envía a Seguridad para que autorice el empleado para entrar en el edificio. Identificar dos Salidas.

APLICACIÓN DE NEGOCIOS EJEMPLO 5 para la regla c): Supongamos un FUR para un programa que combina dos archivos de datos persistentes que describen el mismo objeto de interés, por ejemplo, un archivo de los datos existentes acerca de un objeto de interés 'X' y un archivo con atributos definidos recientemente que describen el mismo objeto de interés 'X'. Identificar dos Lecturas para el proceso funcional, una para cada archivo.

APLICACIÓN DE NEGOCIOS EJEMPLO 6 para la regla c): Supongamos una Lectura de un grupo de datos se requiere en el FUR, pero el desarrollador decide implementarlo por dos comandos para recuperar diferentes subconjuntos de atributos de datos del mismo objeto de interés desde almacenamiento persistente en diferentes puntos el proceso funcional. Identificar una sola Lectura.

APLICACIÓN DE NEGOCIOS EJEMPLO 7 para la regla d): Vea la sección 3.5.2, regla d) para las Entradas, y la sección 3.5.3, la regla b) para las Salidas. (Estos casos se refieren a dos reglas donde las ocurrencias de las Entradas o Salidas tienen sólo un subconjunto de la cantidad máxima de atributos de datos que podría ser movido de acuerdo con la FUR.)

APLICACIÓN DE NEGOCIOS EJEMPLO 8 para la regla d): Supongamos que una Lectura se requiere en el FUR y que en la práctica requiere de muchas ocurrencias de recuperación, como en una búsqueda a través de un archivo. Identificar una sola lectura.

TIEMPO REAL EJEMPLO 9 para la regla d): Supongamos que en un proceso funcional en tiempo real, el FUR requiere que el mismo grupo de datos idénticos deba ser introducido por un usuario funcional dado, por ejemplo, un dispositivo de hardware, dos veces en un intervalo de tiempo fijo con el fin de medir una velocidad de cambio durante el proceso. Los dos movimientos de datos son por lo tanto múltiples ocurrencias de la misma Entrada. Sólo una entrada puede ser identificada para este grupo de datos en este proceso funcional. (No hay manipulación de datos asociado con las dos apariciones de la entrada. El cálculo de la tasa de cambio debe estar asociado a la salida que informa el cambio. Consulte la sección 3.5.6 para los tipos de manipulación de datos que se consideran asociada a una entrada.)

TIEMPO REAL EJEMPLO 10 para las reglas b) y d): Supongamos que un sistema de control de proceso para una máquina que produce un producto plano tal como papel o una película de plástico. La máquina tiene un arreglo de 100 sensores idénticos a través de la dirección de movimiento del producto para detectar roturas o agujeros en el producto, es decir, los sensores deben ser identificables por separado. Sin embargo, el proceso funcional que debe comprobar si hay roturas o agujeros recibe el mismo grupo de datos de cada sensor. Identificar una sola entrada para los datos obtenidos de todos los sensores por el proceso funcional.

3.5.8 Cuando un proceso funcional mueve datos para o desde el almacenamiento persistente

Esta sección explica los movimientos de datos involucrados cuando un proceso funcional de una pieza de software es requerido para obtener algunos datos del almacenamiento persistente, a través de cuatro ejemplos abajo:

- Ejemplo 1 es típico de una aplicación de software, pero podría aplicarse a software en cualquier capa, a excepción de la capa donde el software interactúa directamente con una tienda de hardware. Se requiere que el proceso funcional obtenga datos de almacenamiento persistente pero el FUR no está preocupados por cómo los datos accedidos son manejados por cualquier otro software;
- Ejemplo 2 es para el software con una arquitectura 'cliente-servidor', donde un proceso funcional de un cliente debe solicitar el servidor para algunos datos persistentes;
- Ejemplo 3 se aplica cuando diferentes piezas de software tienen diferentes derechos de acceso a datos persistentes;
- Ejemplo 4 se aplica cuando los datos deben ser obtenidos directamente de un almacenamiento de hardware físico tal vez por el software de controlador de dispositivo.

Los ejemplos se ilustran usando los convenios de Diagramas de Secuencia de Mensajes. La notación de estos diagramas es como se explica a continuación:

- Una flecha vertical hacia abajo representa un proceso funcional
- Las flechas horizontales representan los movimientos de datos, etiquetados como E, X, R o W para la Entrada, Salida, Lectura y Escritura, respectivamente. Entradas y Lecturas se muestran como flechas de llegada al proceso funcional y las Salidas y Escrituras como flechas salientes; aparecen en la secuencia necesaria, de arriba a abajo, del proceso funcional
- Una línea vertical punteada (discontinua) representa un límite frontera

EJEMPLO 1: Cuando un proceso funcional debe mover un grupo de datos hasta o desde un almacenamiento persistente. Este ejemplo se refiere a una pieza de software A que se requiere para recuperar un grupo de datos almacenado en el que el FUR de software 'A' no se refiere a cómo los accesos de datos son manejados por cualquier otro software en las mismas o diferentes capas.

Los usuarios funcionales del software A podrían ser, por ejemplo, los usuarios humanos si el software de A está en la capa de aplicación haciendo una consulta sobre un grupo de datos almacenados. La Figura 3.6 muestra los movimientos de datos COSMIC para esta consulta. La consulta se desencadena por una entrada, seguido de una Lectura del grupo de datos del almacenamiento persistente y luego una salida con el resultado de la consulta. El proceso funcional A no está preocupado por conocer de dónde se recuperan los datos, sólo que los datos son persistentes.

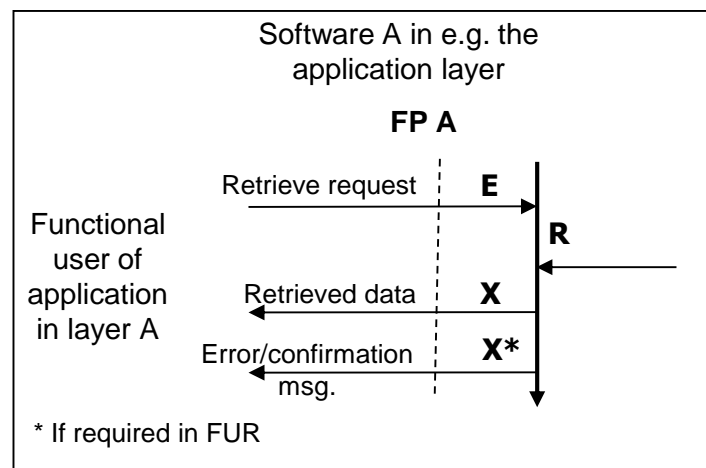


Figura 3.6 - Solución para una Lectura en el software de "A" en la capa de aplicación

Un modelo exactamente análogo se aplicaría si el proceso funcional A fuera requerido para hacer un grupo de datos persistente a través de un movimiento de datos de Escritura. De acuerdo con la regla d) para las condiciones de error (ver sección 3.5.11), los movimientos de datos de Lectura y Escritura se consideran para tener en cuenta cualquier retorno de código o notificación de una condición de error.

La Figura 3.6 muestra un posible mensaje de error específico de la aplicación que podría ser emitido por el proceso funcional A si, por ejemplo, no se encuentra el registro solicitado. Sin embargo, una condición de error no es específico de la aplicación, por ejemplo, 'fallo en el disco' no se cuenta como una salida para el proceso funcional A. Ver también la sección 3.5.11 en los mensajes de error/confirmación.

EJEMPLO 2: Cuando se requiere un proceso funcional para obtener algunos datos de otra pieza de software.

En este ejemplo, las piezas de software a medir se supone que tienen una relación 'cliente / servidor', es decir, cuando una pieza, el cliente, obtiene los servicios y/o datos de la otra pieza, el 'servidor', en la misma o una capa diferente. La Figura 3.7 muestra un ejemplo de una relación, en la que las dos piezas son los principales componentes de la misma aplicación. En cualquier relación de cliente/servidor, el FUR del componente cliente C1 sería identificar el componente servidor C2 como uno de sus usuarios funcionales, y viceversa. Existiría la misma relación y el mismo esquema se

aplicaría si las dos piezas eran aplicaciones por separado, o si una de las piezas fuera un componente de una aplicación separada.

Físicamente, los dos componentes podrían ejecutarse en procesadores separados; en tal caso ellos intercambiarían datos a través de los sistemas operativos respectivos y cualesquiera otras capas intermedias de sus procesadores en una arquitectura de software tal como se muestra en la Figura 2.2. Pero lógicamente, aplicando los modelos COSMIC, los dos componentes intercambian datos a través de una Salida seguida por un movimiento de datos de Entrada. Todo el software y el hardware que interviene se ignoran en este modelo. (Véase también el lado derecho de la Figura 3.1 para un ejemplo similar.).

La Figura 3.7 muestra que un proceso funcional FC1 del componente cliente C1 se desencadena por una entrada de un usuario funcional (como un humano) que consta, por ejemplo, de los parámetros de la consulta. El FUR del componente C1 reconocerá que este componente debe pedir el componente de servidor C2 para los datos requeridos, y debe decirle qué datos son necesarios.

Para obtener el grupo de datos requerido, el proceso funcional FP C1 emite una Salida que contiene los parámetros de la petición consulta al componente C2. Este movimiento de datos de Salida cruza la frontera entre C1 y C2 y así se convierte en la entrada desencadenante de un proceso funcional FP C2 en el componente C2. El proceso funcional FP C2 del componente C2 obtiene los datos necesarios mediante un proceso de Lectura, y envía los datos a C1 mediante una Salida a C1. El proceso funcional FP C1 del componente C1 recibe estos datos como una Entrada. El proceso funcional FP C1 pasa entonces al grupo datos como Salida para satisfacer la consulta de su usuario funcional.

Teniendo en cuenta el posible mensaje de error/confirmación emitida por el cliente, este Ejemplo 2 de consulta requiere 5 movimientos de datos (es decir, 5 CFP) para satisfacer la petición de consulta para el componente C1 y 3 CFP para el componente C2. Esto se compara con el 4 CFP (1 x E, 1 x R y 2 x X) que se habría requerido para el componente C1 si hubiera sido capaz de recuperar el grupo de datos de almacenamiento persistente dentro de su propia frontera a través de una Lectura como se muestra en Figura 3.7.

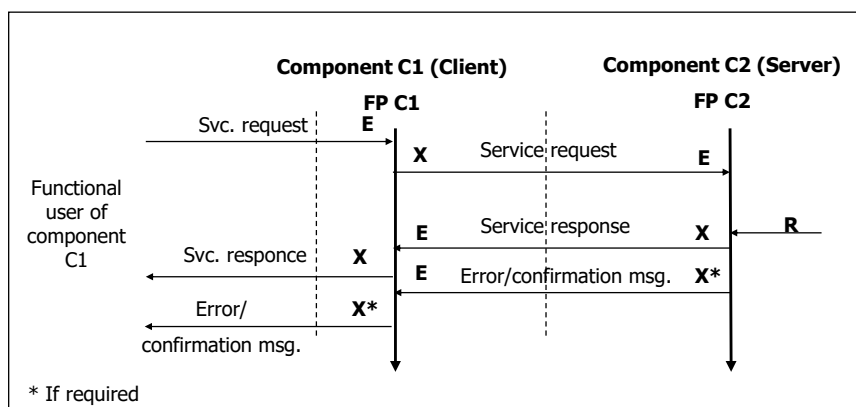


Figura 3.7 - Los intercambios de datos entre los componentes de cliente y servidor

El Componente C2 por supuesto, utilizará los servicios de algún software de controlador de dispositivo de almacenamiento en otra capa de la arquitectura de software para recuperar los datos desde el hardware, como en el Ejemplo 4, Figura 3.9 (b).

Los Ejemplos 1 y 2 ilustran los movimientos de datos cuando se desprende de los FUR que el software se está midiendo debe acceder al almacenamiento persistente dentro de su propio límite, o debe pasar la solicitud de acceso a otra pieza de software fuera de su límite, respectivamente. A veces, sin embargo, la pieza de software que se está midiendo puede tener que utilizar diferentes 'rutas' para acceder a los datos persistentes en función de los atributos de datos específicos para ser accedidos y/o el tipo de acceso (almacenamiento o recuperación). Esto puede surgir cuando el acceso a los datos por el software que se está midiendo está sujeta a reglas o 'derechos' que varía debido a problemas por ejemplo seguridad o privacidad (véase el Ejemplo 3 abajo), o la necesidad de garantizar la integridad de los datos al restringir el acceso de todos a crear, actualizar y eliminar procesos. Cuando la FUR del software que se está midiendo no son claros en este punto, el medidor debe tener cuidado

para determinar los derechos reales de acceso. (No confundir 'derecho de acceso' a los datos con la 'propiedad' de datos, este último es irrelevante para el modelo COSMIC. El almacenamiento persistente no es 'propiedad' de cualquier pieza de software.)

EJEMPLO 3: El software cuenta con diferentes derechos de acceso a los datos almacenados para diferentes propósitos.

Ver Figura 3.8. Una pieza de software A a ser mediar se le permite recuperar ciertos datos almacenados Z (como en el Ejemplo 1, Figura 3.6), pero no se le permite mantener directamente (es decir, crear, actualizar o eliminar) estos mismos datos Z. Cuando el software A necesita mantener los datos Z, el software A tiene que pasar su solicitud a otra pieza de software B a través de una salida seguida de una entrada (análoga a la del ejemplo 2, Figura 3.7 del componente C1 pasando su solicitud al componente C2). La pieza de software B se requiere para asegurar la integridad de los datos Z asegurando una validación consistente, por lo que procesa todas las solicitudes de mantenimiento de datos para la Z.

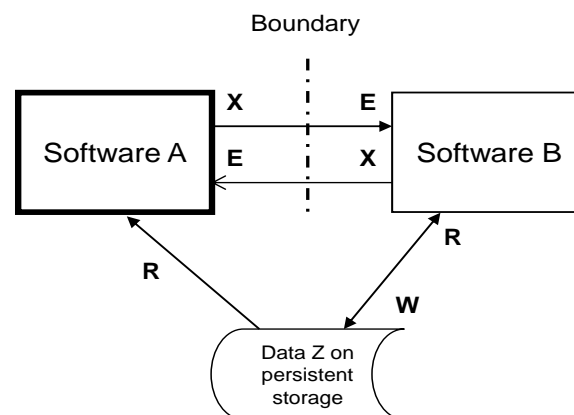


Figura 3.8 - Datos persistente Z dentro de la frontera de ambos programas A y B para una Lectura

En el Ejemplo 3, los modelos COSMIC mostrarían que los datos Z se localizaron en el almacenamiento persistente dentro de los límites de la pieza de software A, pero sólo con fines de recuperación en el que se puede acceder mediante movimientos de datos de Lectura. Para el software B, estos mismos datos Z se localizaron en el almacenamiento persistente dentro de sus límites y el software B puede tanto Leer y Escribir estos datos Z. Para la manipulación de error en este Ejemplo, consulte los ejemplos 1 y 2 anteriores

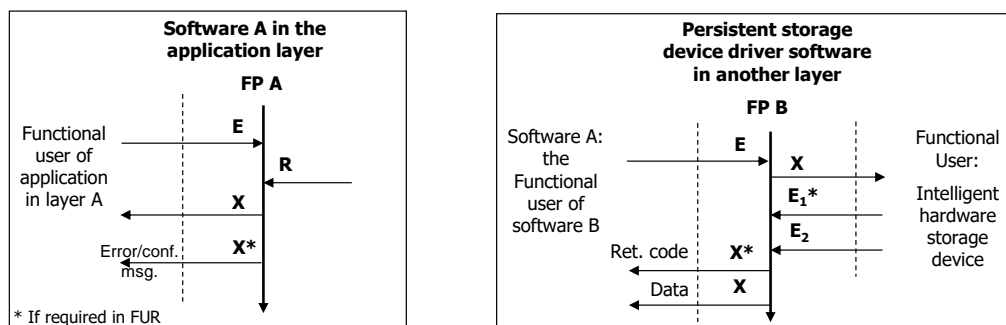
INFRAESTRUCTURA EJEMPLO 4: ¿Cómo son obtenidos los datos persistente por el software de controlador de dispositivo que interactúa con el dispositivo de almacenamiento físico?.

Este ejemplo se refiere a la pieza de software A del Ejemplo 1 que se requiere para recuperar un grupo de datos almacenado. También consideramos una pieza separada del software de 'B' que es el controlador de dispositivo para el almacenamiento de hardware inteligente que mantiene el grupo de datos que la pieza de software de A requiere acceder. (Ignoramos la probable presencia de un sistema operativo para la simplicidad, el sistema operativo transmite con eficacia las solicitudes de aplicación al software de controlador de dispositivo y devuelve los resultados de las solicitudes.)

Las dos piezas de software están en diferentes capas en una arquitectura tal como se muestra en la Figura 2.2. EL software A está por ejemplo en la capa de aplicación, y el software de B está en una capa de controlador de dispositivo. Físicamente, es probable que haya una relación jerárquica entre las dos piezas (ignorando el sistema operativo) y una interfaz física entre el software en las dos capas, como se muestra por ejemplo en la Figura 2.2. Sin embargo, los modelos de los procesos funcionales de software A y B son independientes de la naturaleza de la relación entre las capas, que puede ser jerárquica o bi-direccional.

Los usuarios funcionales del software B en la capa del controlador son la pieza de software A (ignorando el sistema operativo) y el dispositivo de almacenamiento de hardware inteligente que contiene los datos requeridos. ('Inteligente' significa que el dispositivo debe estar en conocimiento de los datos que se necesitan.)

Supongamos que una consulta del proceso funcional FP A del software A necesita recuperar un grupo de datos almacenado. Figura 3.9 (a) muestra el modelo COSMIC de esta consulta. Figura 3.9 (b) muestra el proceso funcional FP B del software B en la capa de controlador de dispositivo que se encarga de la recuperación física de los datos necesarios desde un dispositivo de almacenamiento de hardware (tal como una memoria USB o disco stick).



Figuras 3.9 (a) y (b) - Solución para una Lectura del software de A en la capa de aplicación al software B en la capa de controlador de dispositivo

Figura 3.9 (b) muestra que la solicitud de Lectura del software A se recibe como una Entrada desencadenante para el proceso funcional FP B, que pasa como una Salida para el dispositivo de hardware. La respuesta de este último depende del dispositivo hardware particular. El dispositivo solo puede devolver los datos solicitados, que se muestran como Entrada E_2 en Figura 3.9 b). El dispositivo también puede emitir un mensaje de error por separado que describe el éxito o la razón del fracaso de la solicitud, por ejemplo, 'Datos no encontrados', o 'error de disco', que se muestra como Entrada E_1^* en Figura 3.9 b). FP B devuelve los datos al software de A como una Salida. FP B también normalmente emite un 'código de retorno' que describe el éxito o el motivo del fallo de la solicitud. (Aunque el código de retorno puede estar unido físicamente a los datos devueltos - se trata de datos sobre el resultado del proceso de solicitud). Para FP A no se identifica ninguna Entrada para estos mensajes, como el movimiento de datos de Lectura representa los datos devueltos y los mensajes de error, de acuerdo con la regla d) para una Entrada. Para FP A, una salida se identifica por un mensaje de error/confiración, si es necesario.

Nota: en la práctica, puede haber más movimientos de datos entre el software del controlador del dispositivo y el dispositivo de hardware inteligente que se muestra en la figura 3.8 b). Por ejemplo, esta Figura no muestra el efecto del controlador de dispositivo de medición de un tiempo de espera para la falta de respuesta del hardware.

Comparando los Ejemplos 2 y 4, vemos que en el Ejemplo 4 los modelos de la pieza de software A y el controlador de dispositivo B del Ejemplo 4 no se pueden combinar como lo están en el Ejemplo 2. Esto es porque A y B están en diferentes capas y una Lectura no cruza una frontera. La Figura 3.9 (b) muestra que el software de A es un usuario funcional del software de controlador de dispositivo B. Pero lo contrario no es cierto, porque una Lectura no cruza una frontera. En contraste, la Figura 3.7 muestra los dos componentes en un modelo porque Componente C1 es un usuario funcional del componente C2, y viceversa, y comparten una frontera común.

3.5.9 Cuando un proceso funcional requiere datos de un usuario funcional

Si un proceso funcional debe obtener datos de un usuario funcional hay dos casos. Si el proceso funcional no tiene que decirle al usuario funcional qué datos enviar, una sola entrada es suficiente (por objeto de interés). Si el proceso funcional tiene que decirle al usuario funcional qué datos enviar, una salida seguida de una entrada son necesarias. Las siguientes reglas se aplican:

REGLAS – Cuando un proceso funcional requiere datos de un usuario funcional

- Un proceso funcional deberá obtener un grupo de datos por medio de un movimiento de datos de Entrada de un usuario funcional, *cuando el proceso*

funcional no tiene que decirle al usuario qué datos va a enviar, como en cualquiera de los siguientes cuatro casos:

- Cuando un usuario funcional envía un grupo de datos a través de una Entrada desencadenante que inicia del proceso funcional;
- Cuando un proceso funcional, habiendo recibido un grupo de datos a través de una Entrada desencadenante, espera, expectante el próximo grupo de datos a través de una Entrada desde el usuario funcional (típica en el software de aplicación de negocio, de un usuario funcional humano).
- Cuando un proceso funcional, habiéndose iniciado, pide al usuario funcional 'envíame tus datos ahora, si tienes alguno', y el usuario funcional envía sus datos.
- Cuando un proceso funcional, habiéndose iniciado, inspecciona los datos de un usuario funcional y recupera los datos que necesita.

En estos dos últimos casos (típico de un sistema de votación en tiempo real), por convenio, no habrá una Salida desde el proceso funcional que deba ser identificada para obtener los datos requeridos. El proceso funcional sólo debe enviar un mensaje a un usuario funcional para ingresar sus datos y la funcionalidad del mensaje se considera parte de la entrada. El usuario funcional sabe lo que se puede enviar y el proceso funcional sabe lo que puede esperar. Sólo una entrada es necesaria para este caso.

- b) Cuando un proceso funcional debe obtener los servicios de un usuario funcional (por ejemplo, para obtener datos) y *las necesidades de los usuarios le digan lo que debe enviar* (normalmente cuando el usuario funcional es otra parte del software fuera del alcance del software que se mide), un movimiento de datos de Salida seguida de uno de Entrada serán identificados. La Salida contiene la solicitud específica de los datos; la Entrada contiene los datos.

TIEMPO REAL EJEMPLO 1 de la regla a), tercer o cuarto escenario: Supongamos que un proceso funcional de una aplicación de software de control en tiempo real es requerido para necesario para comprobar un arreglo de sensores idénticos. En el ámbito de aplicación, la solicitud de los datos por el proceso funcional y la recepción de los datos se explica por una Entrada (tipo). (Dado que los sensores son idénticos sólo una entrada (tipo) se identifica y se contaron aunque hay múltiples ocurrencias.)

Supongamos además que la solicitud de los datos debe en la práctica pasar a un dispositivo controlador de software en una capa inferior de la arquitectura, que físicamente obtiene los datos necesarios desde el sensor como se ilustra en la Fig. 2.3. Los procesos funcionales del software de control y de los controladores de software para los sensores sería como se muestra en las Figuras 3.10 (a) and (b) a continuación.

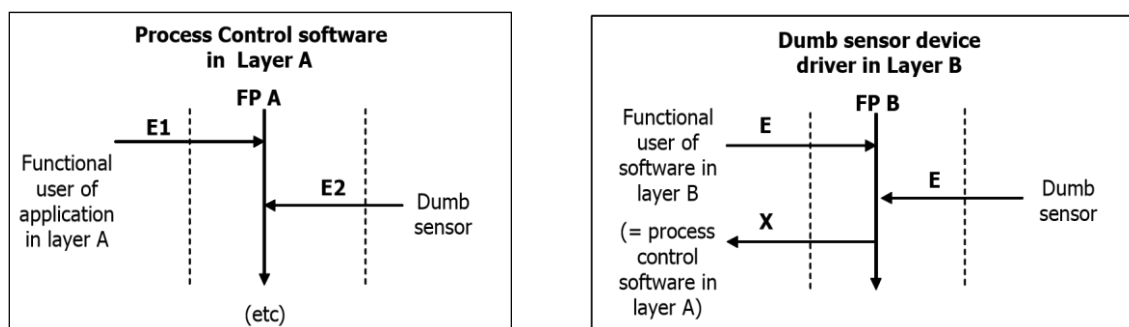


Figura 3.10 (a) y (b) - Solución para una entrada emitida por el software 'A' en la capa de la aplicación de control de proceso recibida por el software 'B' en la capa del dispositivo de control del sensor

La figura 3.10 (a) Muestra que el proceso funcional 'FP A' del software de control de proceso es desencadenado por una entrada 'E1' por ejemplo de una marca de tiempo (tick) de reloj. Después el proceso funcional obtiene datos a través de la entrada 'E2' desde el arreglo de sensores para recibir las múltiples ocurrencias de las lecturas de los sensores. Los sensores son también usuarios funcionales del software de control de proceso en este modelo de nivel de aplicación. (El software del controlador de dispositivo está oculto en este nivel.)

La figura 3.10 (b) Muestra el modelo para el software que acciona los sensores. Recibe los datos a través de una Entrada desde la aplicación (probablemente en la práctica a través de un sistema operativo) como el detonador de un proceso funcional FP B. Este proceso funcional obtiene los datos necesarios a través de una Entrada E de su usuario funcional, el arreglo de sensores.

El grupo de datos se pasa al software de control de procesos a través de una Salida. Esta Salida se recibe como la entrada E2 por el proceso funcional de la aplicación FP A. A continuación el FP, continúa con su procesamiento de los datos del sensor. Una vez más, el hecho de que hay múltiples ocurrencias de este ciclo de recopilación de datos de cada uno de los sensores idénticos, es irrelevante para el modelo.

La aparente desajuste entre una Entrada E2 de un sensor a la aplicación de software de control de procesos y la Entrada seguida de un movimiento de datos de Salida del software de controlador de dispositivo se debe a la convención de que una Entrada de un sensor se considera que incluye cualquier 'solicitud para entrar' en la funcionalidad ya que el usuario funcional no tiene capacidad de manejar algún mensaje de un proceso funcional.

TIEMPO REAL EJEMPLO 2 de la regla b): Suponga que un proceso funcional envía a uno de sus usuarios funcionales como un dispositivo hardware 'inteligente' o a una pieza software semejante algunos parámetros para una consulta o los parámetros de un cálculo, o algunos datos para ser comprimidos. La respuesta del usuario funcional se obtiene a través del proceso funcional mediante una Salida seguida por la recepción de un movimiento de datos de Entrada, como se describe en la sección 3.5.8, Ejemplo 2.

3.5.10 Comandos de navegación y de control de visualización para los usuarios humanos (Comandos de Control')

Un 'comando de control' es un comando que es reconocida en cualquier aplicación que puede ser utilizada por los usuarios funcionales humanos y que deben ser ignorados cuando se mide un tamaño funcional. La definición es:

DEFINICIÓN – Comando de Control
<p>Un comando que le permite al usuario funcional humano controlar el uso del software pero que no involucra ningún movimiento de datos sobre el objeto de interés del software que se está midiendo.</p> <p>NOTA: Un comando de control no es un movimiento de datos porque el comando no mueve datos acerca de un objeto de interés. Ejemplos de ello son comandos de 'página arriba / abajo; presionar la tecla Tab o tecla Enter, hacer clic 'OK' para confirmar una acción anterior, al pulsar un botón para continuar, etc.</p>

REGLA – Comandos de control en el dominio de aplicaciones de negocio
<p>En una aplicación con interfaz para humanos los 'comandos de control' serán ignorados, ya que no implican ningún movimiento de datos sobre un objeto de interés.</p>

EJEMPLOS: Los comandos de control son las funciones que permiten a un usuario funcional controlar la presentación (o no) de un encabezado de una pantalla o los subtotales que han sido

calculados, navegar arriba y abajo y entre pantallas, hacer clic en 'OK' al reconocer un mensaje de error o confirmar algunos datos introducidos, etc. Los Comandos de Control por lo tanto también incluyen comandos del menú que permiten al usuario funcional navegar por uno o más procesos funcionales pero no iniciar algún proceso funcional y comandos para mostrar una pantalla o formulario en blanco para ingresar datos.

Nótese que fuera del ámbito de las aplicaciones con una interfaz de usuario humano, el concepto de una 'comando de control' no tiene un significado especial y cualquier señal o movimiento de datos sobre un objeto de interés procedente de un usuario funcional debe ser considerado, es decir, debe ser medido.

3.5.11 Mensajes de Error/Confirmación y otras indicaciones de condiciones de error

DEFINICIÓN – Mensajes de Error/Confirmación
<p>Una Salida emitida por un proceso funcional a un usuario humano funcional que, o bien confirma solamente que los datos han sido aceptados, o solamente que hay un error en los datos introducidos.</p> <p>NOTA: Cualquier Salida que puede incluir indicaciones de fallo, pero que no está destinado a un usuario funcional humano, no es un mensaje de error / confirmación.</p>

REGLAS – Mensajes de Error/Confirmación y otras indicaciones de condiciones de error
<p>a) Una salida se identificará para contabilizar todos los tipos de mensajes de error/confirmación emitidos por una proceso funcional del software que se mide por todas las causas posibles de acuerdo a su FUR, por ejemplo, éxitos o fracasos de: validación de los datos introducidos o para una llamada para recuperar datos o para hacerlos persistentes, o para la respuesta de un servicio solicitado a otra pieza de software.</p> <p>NOTA: Si los FUR del proceso funcional no requieren ningún tipo de mensaje de error/confirmación a emitir, no identificar ninguna Salida correspondiente.</p> <p>b) Si un mensaje a un usuario funcional humana proporciona datos además de confirmar que los datos introducidos han sido aceptados, o que los datos introducidos es por error, entonces estos datos adicionales deben ser identificados como un grupo de datos movido por una Salida, además de la Salida de error/confirmación.</p> <p>c) Todos los demás datos, emitidos o recibidos por el software que se mide, hacia/desde su hardware o usuarios funcionales software deben ser analizados de acuerdo con los FUR como Salidas o Entradas, respectivamente, de acuerdo con las reglas normales COSMIC, independientemente de si o no los valores de los datos indican una condición de error.</p> <p>d) Las Lecturas y Escrituras se consideran que incluyen cualquier informe de condiciones de error. Por lo tanto ninguna Entrada al proceso funcional que sea mide se debe identificar para cualquier indicación de error recibido como resultado de una Lectura o Escritura de datos persistentes.</p> <p>e) Ninguna Entrada o Salida se identificarán para cualquier mensaje que indica una condición de error que pueda ser emitido, mientras se utiliza el software que se está midiendo, pero que no se requiere que sea procesado en modo alguno por el FUR de ese software, por ejemplo, un mensaje de error emitido por el sistema operativo.</p>

APLICACIÓN DE NEGOCIOS EJEMPLO 1 que ilustra la regla a): En un diálogo humano-computadora, ejemplos de mensajes de error que se producen durante la validación de datos que se introducen podría ser 'error de formato', 'cliente no encontrado', 'Error: por favor, marque la casilla de verificación que indica que ha leído nuestros términos y condiciones', 'límite de crédito superado', etc. Todos estos mensajes de error deben ser considerados como ocurrencias de una Salida en cada proceso funcional donde se producen este tipo de mensajes (que podría ser nombrado 'mensajes de error').

APLICACIÓN DE NEGOCIOS EJEMPLO 2 que ilustra la regla a): Un proceso funcional 'A' potencialmente pueden emitir 2 mensajes de confirmación distintos y 5 mensajes de error a sus usuarios funcionales. Identifique una Salida para dar cuenta de todos estos ($5 + 2 = 7$) Mensajes de error/confirmación. Un proceso funcional 'B' puede potencialmente emitir 8 mensajes de error a sus usuarios funcionales. Identifique una Salida para dar cuenta de estos 8 mensajes de error.

APLICACIÓN DE NEGOCIOS EJEMPLO 3 que ilustra la regla b): En un diálogo humano-computadora, si un mensaje se emite en las situaciones de error pero contiene datos de usuario funcionales, entonces debería ser considerado como una Salida en el proceso funcional donde se produce, además de la Salida de error/confirmación. Un ejemplo de un mensaje de este tipo podría ser 'Advertencia: la cantidad que desea retirar excede su límite de sobregiro en \$ 100' (donde los \$100 es una variable calculada). En este ejemplo, la Salida contiene un grupo de datos sobre la cuenta bancaria del cliente.

APLICACIÓN DE NEGOCIOS EJEMPLO 4 que ilustra la regla e): Emisión de mensajes de error a los usuarios humanos, pero que no son generados o tratados por el software de la aplicación que se está midiendo debe ser completamente ignorado en la medición de la aplicación. Un ejemplo de tal mensaje transmitido desde el sistema operativo podría ser "impresora X no está respondiendo".

TIEMPO REAL EJEMPLO 1 ilustrando la regla c): En un sistema en tiempo real, un proceso funcional que comprueba periódicamente el correcto funcionamiento de todos los dispositivos de hardware que podría emitir un mensaje que informa 'Sensor X ha fallado', donde 'X' es una variable. Este mensaje debe ser identificado como una Salida en ese proceso funcional.

TIEMPO REAL EJEMPLO 2 ilustrando regla c). Los FUR del Ejemplo 1 de TIEMPO REAL en la sección 3.5.9 también puede afirmar que los FP's A y B de la FP deben manejar una condición de error cuando el software de controlador de dispositivo no puede obtener los datos de uno o más sensores de los arreglos. Un sensor no puede, por definición, emitir un mensaje de error. El dispositivo controlador FP B, muy probablemente, obtendrá una cadena de valores del arreglo de sensores, por ejemplo, Estado 1, Estado 2, Estado 3, no hay respuesta, Estado 5, no hay respuesta, Estado 7, etc. y emitirá esta cadena como una Salida del FP A de la aplicación en la que se recibe como una Entrada. No se debe de identificar ningún mensaje de error separado como una salida de FP B del software de controlador de dispositivo, ni como una Entrada a FP A de la aplicación de control de proceso.

LA FASE DE MEDICIÓN

4.0 Resumen del Capítulo

Este capítulo trata de la última etapa del proceso de medición. En primer lugar, la unidad de medida COSMIC se define (un movimiento de datos se mide como un Punto de Función COSMIC, o 'CFP'). A continuación se dan las reglas para asignar un tamaño a los FUR del software que se está midiendo. Las reglas se definen para saber cómo agregar tamaños de diferentes piezas de software.

Además, las normas se definen para saber cómo medir cambios al software que son necesarios (como se maneja en proyectos de 'mejora'). Finalmente, el capítulo discute la posibilidad de 'extensiones locales' para el método COSMIC estándar que pueden utilizarse, por ejemplo, en el ambiente local de una organización que desea contabilizar algunos aspectos de la funcionalidad de una manera que es significativa como un estándar local.

4.1 La fase de medición del proceso

El método general para medir una pieza de software cuando sus Requisitos Funcionales de Usuario se han expresado en términos del Modelo Genérico de Software COSMIC que se resume en la Figura 4.0 a continuación.

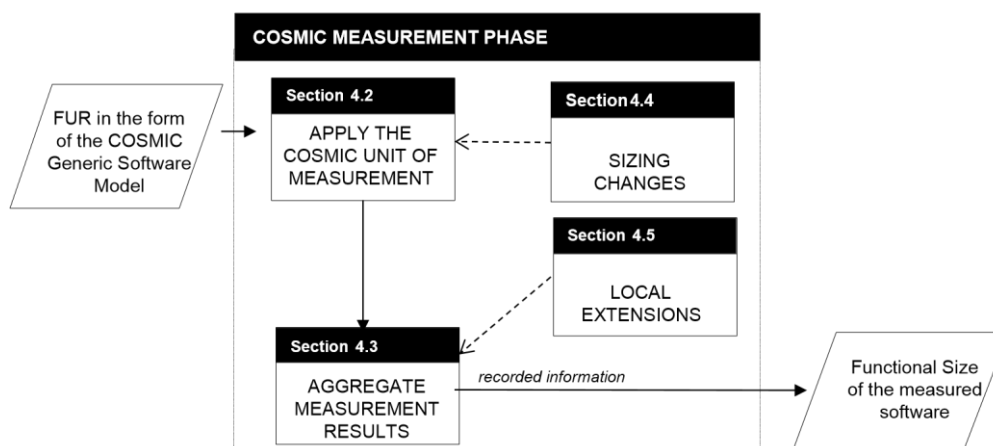


Figura 4.0 – Proceso general de la Fase de Medición COSMIC

Cada paso de este método es el objeto de una sección específica de este capítulo, donde se presentan las definiciones y principios a aplicar, junto con algunas reglas y ejemplos.

4.2 Aplicando la unidad de medida COSMIC

DEFINICIÓN – Unidad de medida COSMIC

1 CFP (Punto de Función COSMIC) que se define como el tamaño de un movimiento de datos.

NOTA: La unidad de medida se conoce como un 'Cfsu', antes de la versión 3.0 del método (unidad de tamaño funcional COSMIC).

A partir de esta definición, cada movimiento de datos (Entrada, Salida, Lectura o Escritura) que se requiera ser añadido, modificado o eliminado por el software que se está midiendo también se mide como 1 CFP.

4.3 Agregando los resultados de medición

Este paso consiste en sumar los resultados de todos los movimientos de datos identificados, en un único valor de tamaño funcional. Este paso se realiza según las siguientes reglas.

4.3.1 Reglas generales de agregación

REGLAS – Agregando los resultados de la medición	
a)	Para cualquier proceso funcional, el tamaño funcional de cada movimiento de datos individual debe ser agregado en un único valor de tamaño funcional en unidades de CFP para luego sumar todos juntos. $\text{Tamaño (proceso funcional)} = \sum \text{tamaño(Entradas}_i) + \sum \text{tamaño(Salidas}_i) + \sum \text{tamaño(Lecturas}_i) + \sum \text{tamaño(Escrituras}_i)$
b)	Para cualquier proceso funcional, el tamaño funcional de los cambios en sus Requisitos Funcionales de Usuario se sumarán al tamaño de movimientos de datos que han sido añadidos, modificados o eliminados en el proceso funcional para dar un tamaño del cambio en unidades de CFP, según la siguiente fórmula. $\text{Tamaño (Cambio(proceso funcional))} = \sum \text{tamaño (movimientos datos añadidos)}_i + \sum \text{tamaño (movimientos datos modificados)}_i + \sum \text{tamaño (movimientos datos eliminados)}_i$ <p>Para más información sobre el agregado del tamaño funcional, véase la sección 4.3.2. Para medir el tamaño de los cambios en el software, véase la sección 4.4.2.</p>
c)	El tamaño de una pieza de software dentro de alcance definido se obtendrá sumando los tamaños de sus procesos funcionales, sujeto a las normas e) y f) a continuación.
d)	El tamaño de cualquier cambio en una pieza de software dentro de un alcance definido se obtendrá sumando los tamaños de todos los cambios de todos los procesos funcionales, sujeto a las normas e) y f) a continuación.
e)	Los Tamaños de piezas de software o de cambios de piezas de software podrán sumarse sólo si se mide al mismo nivel de granularidad de los FUR el proceso funcional.
f)	Los Tamaños de piezas de software y/o cambios en los tamaños de piezas de software dentro de una capa o de diferentes capas serán sumados sólo si tiene sentido hacerlo, a efectos de la medición.
g)	El tamaño de una pieza de software se obtiene sumando los tamaños de sus componentes (independientemente de cómo se descomponen) proporcionados <ul style="list-style-type: none"> • el tamaño de las contribuciones de los movimientos de datos inter-componentes son eliminadas y • sólo una Salida se identifica para todos los mensajes de error/confirmación emitidos por un proceso funcional a un usuario funcional humano.
h)	Si el método COSMIC se extiende localmente (por ejemplo para medir algún aspecto del tamaño no cubierto por el método estándar), entonces el tamaño medido por la extensión local debe ser informado por separado como se describe en la sección 5.1 y no puede ser añadido al tamaño obtenido por el método estándar, medido en CFP (véase además la sección 4.5)

EJEMPLO 1 para las reglas b) y c): Un cambio solicitado para una pieza de software podría ser:

‘añadir un nuevo proceso funcional de tamaño 6 CFP, y en otro proceso funcional agregar un movimiento de datos, hacer modificaciones a otros tres movimientos de datos y eliminar dos movimientos de datos’. El tamaño total del cambio solicitado es de $6 + 1 + 3 + 2 = 12$ CPF.

EJEMPLO 2 para la regla f): Si diversas partes importantes de una pieza de software se desarrollan utilizando tecnologías diferentes, por diferentes sub-equipos de proyecto, no puede haber ningún valor práctico en añadir sus tamaños juntos.

EJEMPLO 3 para la regla g): Si una pieza de software es

- *Primera medición ‘como un todo’, es decir, todo dentro del mismo alcance.*
- *En segundo lugar el tamaño de cada uno de sus componentes se mide por separado, cada uno dentro de su propio ámbito.*

Entonces el tamaño total de la suma del tamaño de todos los componentes por separado (en el segundo caso) superará al tamaño cuando se mide ‘como un todo’ (en el primer caso) debido a la contribución de tamaño de todos los movimientos de datos inter-componentes. Estos movimientos no son visibles cuando la pieza se mide ‘como un todo’. Véase también el ejemplo en la sección sobre la medición en distintos niveles de granularidad en arquitecturas de software puro en el documento ‘Temas avanzados y relacionados’ [12], (para ser re-publicado en la Guía para la aproximación de la medición de tamaño funcional COSMIC’ [6]).

Es de señalar que, dentro de cada capa identificada, la agregación funcional es totalmente escalable. Por lo tanto un sub-total puede ser generado para los distintos procesos funcionales o para todo el software en una capa, dependiendo del propósito y alcance de cada ejercicio de medición y bajo las normas d), e) y f) de arriba.

4.3.2 Más sobre la agregación del tamaño funcional

En un contexto donde el tamaño funcional deba utilizarse como una variable en un modelo, para estimar el esfuerzo por ejemplo, y el software a ser medido se extienda por más de una capa, la agregación normalmente se realizará por capas debido a que el software en diferentes capas a menudo no se implementa con la misma tecnología.

EJEMPLO 1: Considere un software donde la capa de aplicación se ejecutará utilizando 3GL y un conjunto de bibliotecas existentes, mientras la capa de control podría estar implementada utilizando lenguaje ensamblador. El esfuerzo por unidad de tamaño asociada con la construcción del software en cada capa, muy probablemente, sea diferente, y, en consecuencia, una estimación del esfuerzo será preparada por separado para cada capa. Es poco probable que se genere valor en la adición de los tamaños de software en las dos capas.

EJEMPLO 2: Si un equipo de proyecto ha de desarrollar una serie de piezas importantes de software y está interesado en su productividad global, se pueden sumar las horas trabajadas necesarias para desarrollar cada pieza. De manera similar, el equipo puede sumar los tamaños de las principales piezas que ha desarrollado si (pero sólo si) esos tamaños satisfacen las normas dadas anteriormente.

La razón por la que tamaños de piezas de software de diferentes capas de una arquitectura estándar por capas, medida al mismo nivel de granularidad de proceso funcional, puedan sumarse si hace sentido hacerlo () es que esa arquitectura tiene un conjunto coherente definido de usuarios funcionales. El software en cada capa es un usuario funcional del software en otras capas que usa y cualquier pieza de software de una capa puede ser un usuario funcional de cualquier otra pieza de software semejante en la misma capa. Es por lo tanto lógico solamente que los tamaños de las distintas piezas puedan sumarse, siempre sujeto a las normas d), e) y f) arriba indicadas. Sin embargo, en contraste con esto, el tamaño de una pieza de software puede *no* ser obtenida sumando el tamaño de sus componentes reutilizables a menos que el que los movimientos de datos del inter-objeto sean eliminados, por regla f) de arriba.

Agregar los resultados de las mediciones por tipo de los movimientos de datos podría ser útil para analizar la contribución de cada tipo en el tamaño total de un software en una capa y podría ayudar así a caracterizar la naturaleza funcional del software medido en una capa dada.

4.4 Más en la medición del tamaño de los cambios de software

Un ‘cambio funcional’ de un software existente es interpretado en el método COSMIC como ‘cualquier combinación de sumas de nuevos movimientos de datos o de modificaciones o eliminaciones de los

movimientos de datos existentes incluyendo la manipulación de datos asociada'. Los términos 'Mejora' y 'Mantenimiento'⁹ se utilizan a menudo para lo que aquí llamamos un 'cambio funcional'.

La necesidad de un cambio de software puede surgir de cualquier

- Nuevo FUR (es decir, sólo adiciones a la funcionalidad existente), o
- Desde un cambio en el FUR (quizá con sumas, modificaciones y eliminaciones) o
- Desde un 'mantenimiento' necesidad de corregir un defecto.

Las reglas para la medición de cualquiera de estos cambios son las mismas pero el medidor es alertado a distinguir las diversas circunstancias cuando se hacen mediciones del rendimiento y estimaciones.

Cuando un software es reemplazado por completo, por ejemplo al re-escribir, con o sin ampliar y/u omitir funcionalidades, el tamaño funcional de este cambio es el tamaño del software del reemplazo, medido según las reglas normales para la medición del software nuevo. Este caso no será examinado nuevamente en esta sección. El medidor debe ser consciente, sin embargo, de la necesidad que hay cuando se hacen mediciones del rendimiento o estimaciones para distinguir entre los proyectos a desarrollar enteramente con software nuevo y proyectos para 're-hacer o 'reemplazar' software existente.

A menudo, una parte obsoleta de una aplicación es suprimida ('desconectada' sería una mejor descripción) por abandonar el código de programa en lugar de sólo eliminar el contacto con la funcionalidad obsoleta. Cuando la funcionalidad de la parte obsoleta asciende a 100 CFP pero la parte puede ser desconectada por cambiar, digamos, 2 movimientos de datos, 100 y no 2 movimientos de datos serán identificados como el tamaño del cambio funcional. Podemos medir el tamaño de la exigencia, no el tamaño que fue implementado.

Tenga en cuenta que para fines de estimación puede ser aconsejable utilizar una productividad diferente para esta parte del cambio funcional, ya que la desconexión es muy diferente de eliminación 'real'. Alternativamente, para fines de estimación puede ser preferible para medir el tamaño que será implementado (2 CFP en el ejemplo) en lugar del tamaño del requisito (100 CFP en el ejemplo). Si se mide el 'tamaño del proyecto' de 2 CFC, esto debe estar claramente documentado y se distinguen de una medición de los FUR los que requieren que la aplicación deba ser reducida en tamaño por 100 CFC.

Nota: La diferencia entre el tamaño de los cambios funcionales (discutido aquí) y el cambio en el tamaño funcional del software. Generalmente, son diferentes. El tamaño de la última está recogido en la sección 4.4.2.

4.4.1 Modificando funcionalidades

Cualquier movimiento de datos de un determinado tipo (E, X, R y W) incluye dos tipos de funcionalidad: se mueve un solo grupo de datos y tiene algunas manipulaciones de datos asociadas (para el último, véase la sección 3.5.6). Por lo tanto para propósito de la medición un movimiento de datos se considera que es modificado funcionalmente como sigue.

⁹ Una convención de medición normal es que el tamaño funcional de una pieza de software no cambia si el software debe ser cambiado para corregir un defecto con el fin de llevar el software en línea con su FUR. El tamaño funcional del software no cambia si el cambio es para corregir un defecto en la FUR.

DEFINICIÓN – Modificación (de la funcionalidad de un movimiento de datos)

- a) Un movimiento de datos se considera que es modificado funcionalmente si al menos aplica una de las siguientes consideraciones:
- El grupo de datos movido es modificado,
 - La manipulación de datos asociada es modificada.
- b) Un grupo de datos es modificado si al menos aplica una de las siguientes consideraciones:
- Uno o más atributos nuevos son agregados al grupo de datos,
 - Uno o más atributos existentes son removidos del grupo de datos,
 - Uno o más atributos existentes son modificados, por ejemplo, en el significado o en el formato (pero no en sus valores)
- c) Una manipulación de datos se modifica si se cambia funcionalmente de alguna manera.

EJEMPLO: Una manipulación de datos es modificada por ejemplo si ha cambiado el cálculo, el formato específico, presentación y/o validación de datos. 'La presentación' puede significar, por ejemplo la fuente, el color de fondo, la longitud del campo, el número de decimales, etc.

Los comandos de control y los datos de aplicación general de aplicaciones de negocios no implican movimientos de datos, ya que no hay datos sobre los objetos de interés que son movidos. Por lo tanto, cambios de comandos de control y de datos de aplicación general no deben medirse. Por ejemplo, cuando el color de la pantalla para todas las pantallas es cambiado, este cambio no debe medirse. (Ver sección 3.5.10 para una explicación de comandos de control y datos de aplicación general.)

REGLAS – Modificando un movimiento de datos

- a) Si un movimiento de datos debe ser modificado debido a un cambio de la manipulación de los datos relacionados con el movimiento de datos y/o debido a un cambio en el número o tipo de los atributos del grupo de datos movido, un cambio CFP se medirá, independientemente de la cantidad real de modificaciones en el movimiento de la datos.
- b) Si un grupo de datos debe ser modificado, los movimientos de datos moviendo el grupo de datos modificado, cuya funcionalidad no se ve afectada por la modificación del grupo de datos, no serán identificados como movimientos de datos modificados.

NOTA: Una modificación de cualquier dato que figuren en entrada o salida por pantalla que no esté relacionado con el objeto de interés para un usuario funcional no será identificados como un cambio CFP (ver sección 3.3.4 para los ejemplos de estos datos.)

EJEMPLO para la regla a): Una solicitud de cambio para un proceso funcional requiere tres cambios a la manipulación de datos asociada con su Entrada desencadenante y dos cambios a la manipulación asociada con una Salida, así como dos cambios en los atributos del grupo de datos movidos por esta Salida. Medir el tamaño del cambio como 2 CFP, es decir, contar el número total de movimientos de datos cuyos atributos y manipulación de datos asociada debe ser cambiada. NO contar el número de manipulaciones de datos o datos de atributos para ser cambiados.

EJEMPLO para las reglas a) y b): Supongamos un requisito para añadir o modificar los atributos de los datos de un grupo D1, que después de la modificación se vuelve D2. En el proceso funcional 'A' donde esta modificación es necesaria, todos los movimientos de datos afectados por la modificación

deberían ser identificados y contados. Así, según la norma a), si el grupo de datos modificado D2 se hace persistentes y/o es una Salida en el proceso funcional A, identificar una Escritura y/o una Salida de datos respectivamente como modificada. Sin embargo, es posible que haya otros procesos funcionales de Lectura o Entrada sobre este mismo grupo de datos D2, pero su funcionalidad no se ve afectada por la modificación porque estos no utilizan los cambios ni los nuevos atributos de datos. Estos procesos funcionales siguen para procesar el grupo de datos reubicado como si fuese aún D1. Así, según la norma (b), estos movimientos de datos en los otros procesos funcionales que no son afectados por la modificación del movimiento de dato(s) funcional de un proceso funcional A NO deben ser identificados y contados como modificados.

APLICACIÓN DE NEGOCIO EJEMPLO: Si se requiere que un mensaje de error/confirmación sea cambiado (es decir, los textos añadidos, modificados o eliminados) debe ser identificado para la medición, independientemente de si es o no el texto modificado una consecuencia del requisito de cambiar otro movimiento de datos.

4.4.2 Tamaño del software funcionalmente modificado

Después del cambio funcional de una pieza de software, su nuevo tamaño total es igual a:

el tamaño original,

más el tamaño funcional de todos los movimientos de datos agregados,

menos el tamaño funcional de todos los movimientos de datos eliminados

Los movimientos de datos modificados no tienen influencia sobre el tamaño de la pieza de software ya que estos existen tanto antes como después de que las modificaciones han sido realizadas.

4.5 Extendiendo el método de medición COSMIC

4.5.1 Introducción

El método de medición COSMIC de tamaño funcional no pretende medir todos los aspectos del 'tamaño' del software. Así, el método no está actualmente diseñado para medir por separado y de forma explícita el tamaño de los FUR de sub-procesos de manipulación de datos. La influencia sobre tamaño de los sub-procesos de manipulación de datos se toma en cuenta a través de un supuesto simplificador de que es válido para una amplia gama de ámbitos de software, tal como se define en la sección 1.1 sobre la aplicabilidad del método. Además, la influencia del número de atributos de datos por el movimiento de datos no se captura en tamaño del software.

Otros parámetros tales como 'complejidad' (no obstante definido) se podrían considerar que contribuyen al tamaño funcional. Un debate constructivo sobre esta cuestión requeriría primero definiciones consensadas de los otros elementos dentro de la mala noción definida de 'tamaño' como se aplica al software. Tales definiciones son todavía, en este punto, objeto de investigación y de mucho debate.

Sin embargo, la medida de tamaño COSMIC se considera una buena aproximación para el estado del método propuesto y el ámbito de aplicabilidad. Sin embargo, puede ser que dentro del entorno local de una organización que utilice el método de medición COSMIC, se desee para tener en cuenta esa funcionalidad en una forma que sea válida como un estándar local. Por esta razón, el método de medición COSMIC permite extensiones locales. Cuando esas extensiones locales son utilizadas, los resultados de medición deben ser comunicados según la convención especial presentada en la sección 5.1. Las siguientes secciones muestran cómo extender el método con un estándar local.

4.5.2 Manipulación de software rico en datos

El método COSMIC fue diseñado para medir software 'rico en movimiento de datos'. Al igual que todos los demás métodos verdaderos de Medición de Tamaño Funcional (FSM), no fue diseñado para medir de forma explícita la funcionalidad de manipulación de datos. En cambio, el método asume que los tipos de movimiento de datos contabilizan la funcionalidad de manipulación de datos asociados (véase más adelante). Esta suposición ha demostrado ser razonable para todos los propósitos prácticos, tales como medición de desempeño del proyecto y la estimación para lo que el método fue diseñado y para los ámbitos en los que se utiliza comúnmente.

La experiencia ha demostrado sin embargo que el método puede también ser aplicado con éxito a menudo a tamaño software 'rico en manipulación de datos', por ejemplo, algunos software científico/ingeniería. Esto es cierto, por ejemplo, donde el software debe manejar grandes volúmenes de datos, dando lugar a un gran número de tipos de movimiento de datos. Este último puede explicar de manera efectiva para cualquier manipulación de datos matemáticamente-compleja que también pueden estar presentes. Por 'aplicado con éxito', queremos decir que el método ha producido tamaños significativos y útiles en relación con el objeto de la medición. Los ejemplos incluyen el dimensionamiento de sistemas expertos, software para procesar digitalmente las variables continuas, un software que recopila y analiza los datos de los experimentos científicos o de medidas de ingeniería, etc.

Sin embargo, dado el diseño fundamental del método COSMIC, los usuarios del método, cuando se enfrentan a la medición de un tamaño funcional del software que es rico en la manipulación de datos, debe decidir por sí mismos si el método realmente produce tamaños funcionales que son significativos y útiles en relación para el propósito de la medición. Donde el método no puede contabilizar adecuadamente para manipulación de datos, puede ser posible desarrollar una extensión local del método para superar la limitación - véase la sección 4.5.5.

4.5.3 Limitaciones de los factores que contribuyen al tamaño funcional

Dentro de sus ámbitos de aplicación, el método COSMIC no intenta medir todos los aspectos posibles de la funcionalidad que pueda ser considerado para contribuir al 'tamaño' del software. Por ejemplo, el método de medición no captura explícitamente la influencia de la 'complejidad' de software. Pero hay muchos tipos de complejidad, por ejemplo, arquitectónico, semántica, de tiempo, proceso, datos, etc., y en la medición de tamaño funcional, el método actualmente contabiliza de una manera sencilla la contribución al tamaño de la complejidad del proceso (y por tanto indirectamente de complejidad de datos).

El método también no considera la influencia del número de atributos de datos por el movimiento de datos en el tamaño funcional de software. Como se describe en la sección 4.5.6, si se desea, tales aspectos de tamaño funcional pueden ser apoyados por una extensión local para el método de medición COSMIC.

4.5.4 Limitaciones en la medición de piezas de software muy pequeñas

Todos los métodos de medición de tamaño funcional se basan en los supuestos de un modelo simplificado de la funcionalidad del software que se pretende que sea razonable "en promedio" para el ámbito de intención de la aplicabilidad y la utiliza en la medición del desempeño del proyecto y estimación. Por lo tanto, es necesario tener precaución al medir, comparar o usar tamaños de piezas de software muy pequeñas, y especialmente de los cambios muy pequeños en una pieza de software, donde el supuesto de 'promedio' puede romperse. En el caso del método COSMIC, 'muy pequeño' significa 'pocos movimientos de datos'.

4.5.5 Extensiones locales con algoritmos complejos

Si se considera necesario para tener en cuenta algoritmos complejos, puede organizarse un estándar local para esta funcionalidad excepcional. En cualquier proceso funcional donde existe una manipulación de datos anormalmente compleja de un sub-proceso funcional, el medidor está libre de asignar su propia determinación-local de puntos de función.

EJEMPLO: Una extensión local estándar puede ser: "En nuestra organización, un punto de función local FP se asigna a algoritmos matemáticos como (lista de definiciones locales y ejemplos bien entendidos). Dos puntos de función locales FP's son asignados para (otra lista de ejemplos), etc."

4.5.6 Extensión Local con sub-unidades de medida

Cuando se necesita más precisión en la medición de los movimientos de datos, entonces se puede definir una sub-unidad de medida. Por ejemplo, un metro puede ser sub-dividido en 100 centímetros o 1000 milímetros. Análogamente, el movimiento de un único atributo podría ser utilizado como subunidad de medida. Las mediciones sobre una pequeña muestra de software en los ensayos de campo de COSMIC indican que en la muestra medida, el número promedio de atributos de datos por movimiento de datos no varía mucho en los cuatro tipos de movimiento de datos. Por esta razón y por motivos de facilitar las mediciones, la unidad de medida de COSMIC, 1 CFP, ha sido fijada en el nivel de un

movimiento de datos. Sin embargo, la precaución es claramente necesaria cuando se comparan los tamaños medidos en CFP (puntos de función COSMIC) de dos piezas de software distintas donde el número promedio de atributos datos por movimiento difiere notablemente entre ambas.

Cualquiera que desee refinar el método COSMIC al introducir una sub-unidad de medida es libre de hacerlo, pero debe dejar claro que el tamaño resultante medido no se expresa en el estándar de puntos de función COSMIC.

INFORME DE MEDIDAS

5.0 Resumen del Capítulo

Cuando una medición está terminada y aceptada, el resultado debe ser reportado y datos sobre la medición archivada a fin de garantizar que el resultado es siempre inequívocamente interpretable. El capítulo lista los parámetros que se deben considerar para el registro.

5.1 Etiquetado

El Modelo Genérico de Software se puede representar en forma de matriz donde las filas representan los procesos funcionales (que podrían ser agrupados por capas), las columnas representan grupos de datos y las celdas tienen los subprocesos identificados (Entrada, Salida, Lectura y Escritura). Esta representación del Modelo Genérico de Software se presenta en el Apéndice A.

Los resultados de la medición COSMIC deben ser reportados y archivados de acuerdo a las siguientes convenciones. Al informar de un tamaño funcional COSMIC se debería etiquetar de acuerdo a la siguiente regla, relacionada con el estándar ISO/IEC 14143-1:2007.

REGLA – Etiquetado de las mediciones COSMIC

Los resultados de una medición COSMIC se deberían anotar como ' x CFP($v.y$)', donde:

- ' x ' representa el valor numérico del tamaño funcional
- ' $v.y$ ' representa la identificación de la versión estándar del método COSMIC usado para obtener el valor numérico del tamaño funcional " x ".

NOTA: Si un método de aproximación local se usó para obtener la medida, pero aparte la medida se hizo utilizando las normas de la versión estándar COSMIC, la regla de etiquetado anterior debería utilizarse, pero en algún sitio se deberá anotar el uso del método de aproximación.-ver sección 5.2.

EJEMPLO: Un resultado obtenido empleando las reglas de este Manual de Medición se debería nombrar como ' x CFP ($v4.0.1$)'

Cuando se usan extensiones locales, como se define en la sección 4.5, el resultado de la medida debe ser informado como se define a continuación:

REGLA – Etiquetado de extensiones locales COSMIC

Un resultado de una medición COSMIC utilizando extensiones locales debería ser anotado como:

x CFP ($v. y$) + z Local FP', donde:

- ' x ' representa el valor numérico obtenido al agregar todos los resultados de medidas individuales de acuerdo al método estándar COSMIC, versión $v.y$.
- ' $v.y$ ' representa la identificación de la versión estándar del método COSMIC usado para obtener el valor numérico de tamaño funcional " x ".
- ' z ' representa el valor numérico obtenido al agregar todos los resultados individuales de medida obtenidos de las extensiones locales al método COSMIC.

5.2 Archivando los resultados de medición COSMIC

Al archivar los resultados de medición COSMIC, se debe mantener la siguiente información para asegurar que el resultado sea siempre interpretable.

REGLA – Reportes de mediciones COSMIC

Además de las medidas reales, descritas en el punto 5.1, todos o algunos de los siguientes atributos de cada medición se debe registrar, dependiendo del propósito de medición y el nivel deseado de comparabilidad con otras medidas, por ejemplo, para fines de evaluación comparativa.

- a. Identificación del componente de software medido (nombre, versión ID o Configuración ID)
- b. Las fuentes de información usadas para identificar el FUR utilizado para la medición.
- c. El ámbito del software.
- d. Una descripción de la arquitectura en capas en las que la medición es realizada, si aplica.
- e. Una especificación del propósito de la medición.
- f. Una descripción del alcance de la medición y su relación con el alcance global de una serie de medidas relacionadas, si las hubiera. (Usar las categorías de alcance genéricas de la sección 2.2)
- g. El patrón de medida utilizado (COSMIC o local), con el modo de procesamiento (online o por lotes).
- h. Los usuarios funcionales del software
- i. El nivel de granularidad de los artefactos de software disponibles y el nivel de descomposición del software.
- j. El punto en el ciclo de vida del proyecto cuando se hizo la medición (especialmente si la medida es una estimación basada en FUR incompletas, o si en realidad se hizo sobre la bases de funcionalidad entregada).
- k. El objetivo o margen de error que se considera para la medida
- l. Indicaciones de si el método de medición estándar COSMIC fue utilizado, y/o se usó una aproximación local al método estándar y/o se utilizaron extensiones locales (ver sección 4.5). Usar las reglas de etiquetado de las secciones 5.1 y 5.2.
- m. Una indicación de si la medida es de funcionalidad desarrollada o entregada (funcionalidad 'desarrollada' se obtiene creando nuevo software; funcionalidad 'entregada' incluye funcionalidad desarrollada y también incluye funcionalidad obtenida por otros medios diferentes a crear nuevo software, p.ej: incluir todas las formas de reutilización de software existente, el uso de parámetros existentes para añadir o cambiar funcionalidad, etc.).
- n. Una indicación de si la medida es de funcionalidad recientemente proporcionada o si es el resultado de una actividad de 'mejora' (p.ej: la suma es de funcionalidad añadida, cambiada o borrada-ver 4.4)
- o. El número de componentes importantes, si es aplicable, cuyos tamaños han sido añadidos para el tamaño total registrado.
- p. El porcentaje de funcionalidad implementada por el software de re-utilización.
- q. Para cada alcance dentro del alcance global de la medición, una matriz de medida, como se especifica en el apéndice A
- r. El nombre del medidor y cualquier certificación de COSMIC, la fecha de la medición.

REFERENCES

Todos los documentos COSMIC que se indican a continuación, incluyendo traducciones a otros idiomas, se pueden encontrar en www.cosmic-sizing.org.

Los títulos de los documentos COSMIC no dan el número de versión método al que se refieren. Todos los documentos están siendo actualizados para que estén en línea con la versión 4.0.1 del método.

- [1] ISO/IEC 19761:2011 Software Engineering – COSMIC: a functional size measurement method, www.iso.org
- [2] Introduction to the COSMIC Method of measuring software
- [3] (Example of several papers by the same authors) Al-Sarayreh, K.T. and A. Abran, Specification and Measurement of System Configuration Non Functional Requirements, 20th International Workshop on Software Measurement (IWSM 2010), Stuttgart, Germany, 2010
- [4] Guideline for Sizing Real-time Software
- [5] Guideline for 'Measurement Strategy Patterns
- [6] Guideline for approximate COSMIC functional size measurement (under construction)
- [7] Guideline for Sizing Business Application Software
- [8] Guideline for sizing Data Warehouse Application Software
- [9] Guideline for Sizing Service-Oriented Architecture Software
- [10] Quick Reference Guide to the COSMIC method for sizing Business Application Software
- [11] Quick Reference Guide to the COSMIC method for sizing Real-Time Application Software
- [12] Advanced and Related Topics
- [13] Guideline for Convertibility (under construction)
- [14] International Vocabulary of Basic and General Terms in Metrology, International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1
- [15] Adapted from Merriam Webster's Collegiate Dictionary, 10th Edition
- [16] Adapted from Merriam Webster's Collegiate Dictionary, 10th Edition, and La Petit Larousse Illustré, 1996 Edition
- [17] ISO/IEC 14143/1:2011 Information technology – software measurement – functional size measurement. Part 1 Definition of concepts

- Para cada proceso funcional identificado, los movimientos de datos se suman por tipo y cada total se registra en la columna correspondiente en el extremo derecho de la matriz
- El resumen de medición puede entonces ser calculado y registrado en las celdas de cada componente, en la línea 'Total'.

APÉNDICE B – EVOLUCIÓN DE LOS REQUISITOS NO FUNCIONALES – EJEMPLOS

La siguiente tabla muestra algunos ejemplos de los requisitos de declaraciones que pueden aparecer inicialmente a nivel de sistema (incluso antes de que los requisitos se han asignado a software o hardware) o en el nivel de software como no funcionales que evolucionan, en todo o en parte, como un proyecto que avanza, en una mezcla de FUR para el software y declaraciones de los requisitos que son verdaderamente "no funcionales".

Columna 1: ejemplos de declaraciones de sistema o software NFR

Columna 2: ejemplos de FUR software que pudiera resultar, como un progreso de proyecto, de NFR en la columna 1. Los FUR puede ser para el software que se desarrolle o para ser adquirido por ejemplo, 'COTS' software (Commercial Off-The-Shelf).


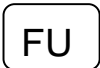
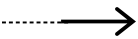

Columna 3: ejemplos de los requisitos y limitaciones en el sistema o proyecto que pudiera quedan después de la separación de los FUR de software en la columna 2. Se trata, por lo tanto, de los verdaderos requisitos 'no funcionales'.

Requisitos de Sistema o de software que en un principio pueden aparecer como no funcionales	Ejemplos de FUR para el software, que será desarrollado o adquirido, que puede evolucionar a partir de los NFR iniciales de sistema	Ejemplos de verdaderos NFR que puedan permanecer después de algunos requisitos del sistema han convertido en FUR de software
El tiempo de respuesta del sistema durante las horas pico no excederá un promedio de X segundos.	Software para: <ul style="list-style-type: none"> • Introducir los datos externos necesarios por el sistema en tiempo real • Monitorear e informar sobre el tiempo medio de respuesta 	Hardware específico (rápido) Algunos programas de software que se escribirán en un lenguaje de bajo nivel La definición objetivo de tiempo de respuesta
La disponibilidad del sistema excederá Y% como promedio durante cada año natural	Software que permite el cambio rápido de procesamiento a un procesador de respaldo sin interrupción de servicio	Procesador de respaldo de hardware que funciona en modo "hot standby" La definición objetivo de disponibilidad
Los parámetros de aplicación deberán ser fácilmente mantenible por personal de usuario	Software que permitir a los usuarios mantener las tablas de parámetros	Ninguna
El sistema deberá ser utilizable por los miembros del público en general que no tienen la formación y la finalización con éxito se considera con un porcentaje de Z%	Software para: <ul style="list-style-type: none"> • brindar facilidades de ayuda integral • ofrecer menús bien estructurados para facilitar su uso • apoyo a los usuarios con visión parcial 	<ul style="list-style-type: none"> • Requisitos para teclados braille • Numerosas pruebas de los miembros del público en general • La proporción Z% objetivo
El usuario tendrá la opción de asegurar los archivos por cifrado	Software para cifrar y descifrar archivos sobre la demanda del usuario	El uso de un hardware 'dongle' o dispositivo clave de cifrado
El sistema deberá ser portable a través de X, Y y Z entornos de hardware/software	Una capa de software para aislar la funcionalidad principal de los requisitos de interfaz específicas de los ambientes X, Y y Z	El uso de un lenguaje altamente portátil como Java

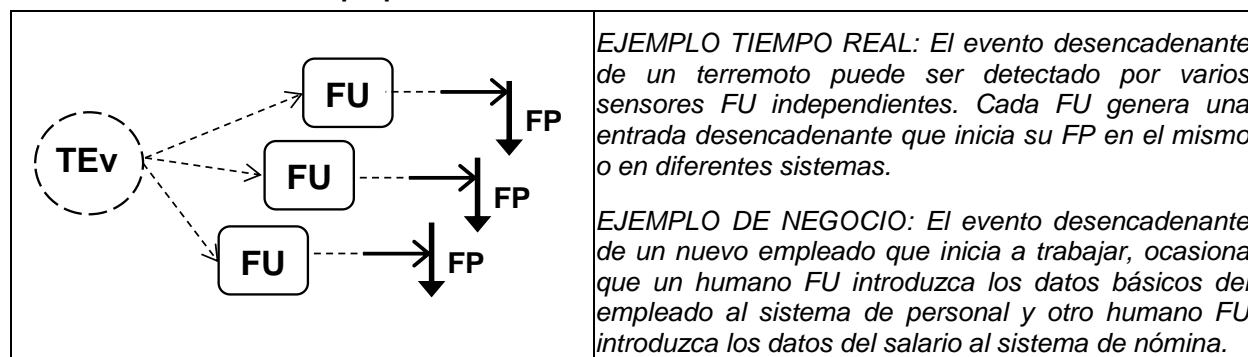
APÉNDICE C - CARDINALIDAD DE EVENTOS DESENCADENANTES, USUARIOS FUNCIONALES Y PROCESOS FUNCIONALES

Todas las relaciones a lo largo del evento desencadenante / usuario funcional / Entrada desencadenante / cadena del proceso funcional (como se muestra en la Figura 3.3) pueden ser de muchos a muchos en principio, con una excepción. (La excepción es que cualquier entrada desencadenante puede iniciar sólo un proceso funcional - véase la regla b) para un proceso funcional en la sección 3.2.2).

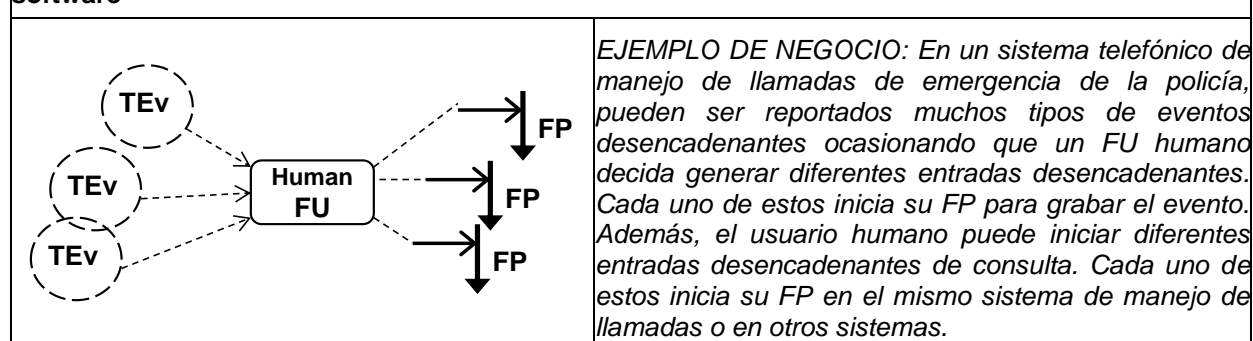
La siguiente tabla muestra ejemplos de posibles relaciones. Tenga en cuenta que los casos pueden no ser exhaustivos. La tabla utiliza las siguientes abreviaturas y

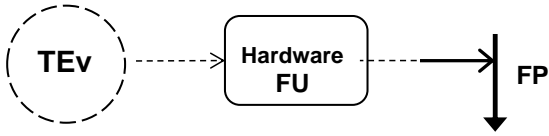
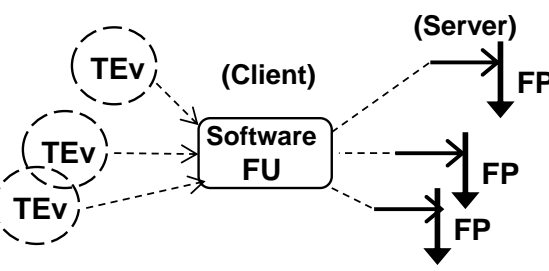
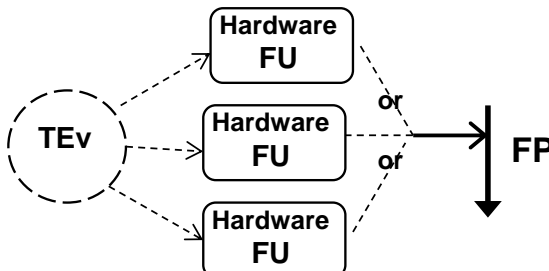
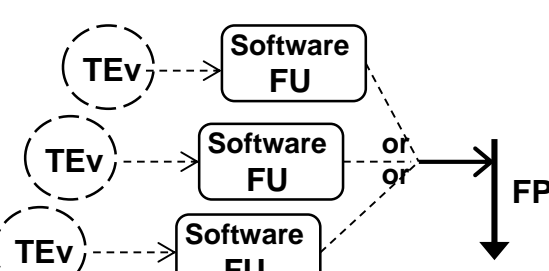
	Evento desencadenante		Usuario Funcional
	Grupo de datos (parte punteada) movido por una Entrada desencadenante (flecha sólida)		Proceso Funcional

1. Un único evento desencadenante puede ocasionar que múltiples FU generen una entrada de desencadenante en el mismo o en diferentes sistemas de software. Cada entrada de desencadenante inicia su propio FP

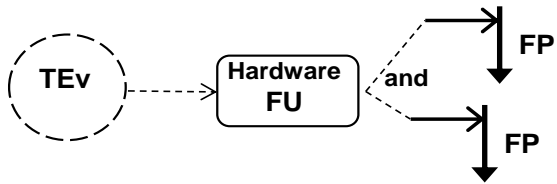


2. Cada evento desencadenante ocasiona que un humano FU genere una Entrada desencadenante diferente. Cada entrada desencadenante inicia su FP en el mismo o en diferentes sistemas de software



<p>3. A FU hardware o software puede ser diseñado para detectar (o 'generar') uno o más tipo (s) específico (s) de eventos. Cada uno de ellos ocasiona que el FU genere una entrada desencadenante. Cada uno de estos inicia su FP en el mismo sistema de software.</p>	
	<p>EJEMPLO TIEMPO REAL: Cuando la temperatura del líquido que alcanza un nivel preestablecido (el evento desencadenante), un temporizador FU genera una entrada desencadenante para comenzar su FP en un sistema de software específico.</p>
	<p>EJEMPLO DE NEGOCIO: En una aplicación de software distribuido, el componente de cliente es un FU del componente de servidor. Diferentes necesidades de información (los eventos desencadenantes) del componente cliente ocasionan que se generen diferentes entradas desencadenantes, cada uno inicia el FP del componente de servidor, para cada tipo de servicio que necesita.</p>
<p>4. Dos o más hardware FU del mismo software puede identificar el mismo evento desencadenante. Cada FU puede iniciar la entrada desencadenante que inicia el mismo FP.</p>	
	<p><i>EJEMPLO TIEMPO REAL: El evento desencadenante de una situación anormal en un sistema de control de proceso en tiempo real, puede ser detectada por uno o más componentes de hardware FU. Cada FU puede iniciar la parada de emergencia FP.</i></p> <p>(NOTA: Cualquier ocurrencia de este FP será iniciado por el primer FU que detecte el evento desencadenante).</p>
<p>5. Dos o más FU pueden iniciar cada uno una entrada desencadenante que inicia el mismo FP</p>	
	<p><i>EJEMPLO DE INFRAESTRUCTURA: Varios FU de software pueden cada uno 'llamar', por ejemplo: iniciar el mismo FP en el mismo componente de software reutilizable. (En este caso el FU de software 'genera' el evento cuando llama al componente.)</i></p> <p>(NOTA: Cualquier ocurrencia de este FP puede ser iniciada solamente por uno de los FU de software posibles en un cierto momento)</p>

6. En la detección de un evento desencadenante, un FU podría iniciar dos o más Entradas desencadenantes. Cada entrada desencadenante inicia su FP en el mismo software



EJEMPLO TIEMPO REAL: En un sistema de control doble de seguridad crítica, un evento desencadenante puede ocasionar que un FU (por lo general de hardware) genere dos entradas desencadenantes, cada una de ellas iniciando su FP. Los dos FP podrían, por ejemplo, tener la misma FUR pero ser desarrollada por grupos separados, como resultado de una estrategia de diversidad.

APÉNDICE D – RESUMEN DE PRINCIPIOS Y REGLAS DEL MÉTODO COSMIC

La siguiente tabla identifica cada principio y la regla encontrada en la v4.0.1 del método de medición COSMIC, con el propósito de referenciar precisamente, con el número de sección en la columna izquierda.

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
1.3.1	<p>Modelo Contextual de Software</p> <p>Principios</p> <ul style="list-style-type: none"> a) El software está limitado por el hardware. b) El software está típicamente estructurado en capas. c) Una capa puede contener una o más aplicaciones de software semejante d) Cualquier aplicación software que deba medirse, se define por su alcance de medición, que se limitará en su totalidad dentro de una sola capa. e) El alcance de la aplicación de software debe medirse en función de la finalidad de la medición. f) Los usuarios funcionales de una aplicación de software se identificarán a partir de los FUR de la aplicación software que se medirá como los emisores y/o destinatarios de los datos al/desde el software respectivamente. g) Los FUR de software pueden expresarse en distintos niveles de granularidad. h) Una medición precisa en COSMIC del tamaño de una pieza de software requiere que sus FUR sean conocidos en un nivel de granularidad en el que se puedan identificar sus procesos funcionales y subprocesos. i) Una medición aproximada en COSMIC de una pieza de software es posible si sus FUR se miden a un alto nivel de granularidad por un enfoque de aproximación y es escalado al nivel de granularidad de procesos funcionales y subprocesos.
1.3.2	<p>Modelo Genérico de Software</p> <p>Principios</p> <ul style="list-style-type: none"> a) Una pieza de software interactúa con sus usuarios funcionales a través de una frontera, y con un almacenamiento persistente que existe dentro de esa frontera. b) Los requisitos funcionales de los usuarios de la aplicación software a medir pueden representarse como procesos funcionales únicos. c) Cada proceso funcional se conforma por sub-procesos. d) Un sub-proceso puede ser un movimiento de datos o una manipulación de datos. e) Un movimiento de datos mueve un sólo grupo de datos. f) Existen cuatro tipos de movimientos de datos: <ul style="list-style-type: none"> • Una Entrada, mueve un grupo de datos hacia un proceso funcional desde un usuario funcional. • Una Salida mueve un grupo de datos fuera de un proceso funcional hacia un usuario funcional. • Una Escritura mueve un grupo de datos desde un proceso funcional hacia un almacén persistente.

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<ul style="list-style-type: none"> • Una Lectura mueve un grupo de datos desde un almacén persistente hacia un proceso funcional <p>g) Un grupo de datos consiste en un conjunto único de atributos de datos que describen un único objeto de interés.</p> <p>h) Cada proceso funcional es activado por un movimiento desencadenante de entrada de datos. El grupo de datos movido por el evento desencadenante es generado por un usuario funcional en respuesta a un evento desencadenante.</p> <p>i) Un proceso funcional deberá incluir al menos un movimiento de entrada y también un movimiento de Salida o Escritura de datos, es decir que deberá incluir un mínimo de dos movimientos de datos. No existe un límite superior para el número de movimientos de datos en un proceso funcional.</p> <p>j) Como una aproximación para fines de medición, los sub-procesos de manipulación de datos no se miden por separado; la funcionalidad de cualquier manipulación de datos se supone que se tomada en cuenta por el movimiento de datos con el que está asociado</p>
1.4	<p>El Principio de Medición COSMIC</p> <p>Principios</p> <p>a) El tamaño de un proceso funcional es igual al número de sus movimientos de datos.</p> <p>b) El tamaño funcional de una pieza de software de alcance definido es igual a la suma de los tamaños de sus procesos funcionales.</p>
2.2	<p>Alcance de la medición</p> <p>Reglas</p> <p>a) El alcance de cualquier pieza de software a ser medida se deriva del propósito de la medición.</p> <p>b) El alcance de cualquier medición no se extenderá por más de una capa de software que se desea medir.</p>
2.2.2	<p>Capa</p> <p>Principios</p> <p>a) El software en una capa proporciona un conjunto de servicios que es coherente de acuerdo con algún criterio definido, y ese software puede utilizarse en otras capas sin saber cómo se implementan estos servicios.</p> <p>b) La relación entre el software en cualquiera de dos capas se define por una 'regla de correspondencia' que puede ser</p> <ul style="list-style-type: none"> • 'jerárquica', es decir, el software en la capa A tiene permitido utilizar los servicios proporcionados por el software en la capa B, pero no al revés (donde la relación jerárquica puede ser hacia arriba o hacia abajo), o • 'bidireccional', es decir, el software en la capa A tiene permitido utilizar el software en la capa B, y viceversa. <p>c) El Software en una capa intercambia grupos de datos con el software en otra capa a través de sus respectivos procesos funcionales.</p> <p>d) El Software en una capa no necesariamente utiliza todos los servicios funcionales proporcionados un software en otra capa.</p> <p>e) El Software en una capa de una arquitectura de software definida puede ser dividido en otras capas de acuerdo a diferentes arquitecturas de software definidas.</p>

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
2.3.1	<p>Usuarios Funcionales</p> <p>Reglas</p> <ul style="list-style-type: none"> a) Los usuarios funcionales de una pieza de software a ser medida, deben ser derivados del propósito de la medición b) Para los usuarios funcionales que son funcionalmente idénticos según los FUR, se identifica un solo tipo de usuario funcional. c) Cuando el propósito de una medición de una pieza de software está relacionado con el esfuerzo para desarrollar o modificar el software, entonces los usuarios funcionales deben ser todos aquellos emisores y/o receptores de datos hacia/desde la nueva funcionalidad o la modificada, como es requerido por sus FUR.
2.4.3	<p>Nivel de Granularidad de un Proceso funcional</p> <p>Reglas</p> <ul style="list-style-type: none"> a) Una medición precisa de tamaño funcional de una pieza de software requiere que sus FUR sean conocidos a nivel de granularidad en la que se pueden identificar sus procesos funcionales y sub-procesos de movimiento de datos. b) Si algunos requisitos deben medirse antes de que se hayan definido con suficiente detalle para una medición precisa, los requisitos se pueden medir utilizando un enfoque de aproximación. Estos enfoques definen cómo los requisitos pueden ser medidos en niveles de granularidad superiores. Los factores de escala se aplican entonces a las mediciones realizadas a niveles de granularidad más altos para producir un tamaño aproximado al nivel de granularidad de procesos funcionales y sus subprocesos de movimientos de datos. Consulte la ' Guía para la aproximación de la medición de tamaño funcional COSMIC ' [6].
3.2.2	<p>Proceso Funcional</p> <p>Reglas</p> <ul style="list-style-type: none"> a) Un proceso funcional pertenecerá íntegramente a la alcance de la medición de una pieza de software en una, y sólo una, capa. b) Cualquiera Entrada desencadenante de una pieza de software que se está midiendo puede iniciar sólo un proceso funcional en ese software. c) Un proceso funcional comprenderá al menos dos movimientos de datos, una Entrada y además una Salida o una Escritura. No hay límite superior para el número de movimientos de datos en un proceso funcional. d) Un proceso funcional en ejecución se debe considerar terminado cuando se ha satisfecho su FUR para la respuesta a su Entrada desencadenante. Una pausa durante la tramitación por razones técnicas no se considerará como la terminación del proceso funcional.
3.3.1	<p>Grupo de Datos</p> <p>Principio</p> <p>Cada grupo de datos identificado deberá ser único y distinguible a través de su colección única de atributos de datos.</p>
3.5.2	<p>Entrada, (Entry, E) Principios</p> <ul style="list-style-type: none"> a) Una Entrada deberá mover un único grupo de datos describiendo un solo objeto de interés de un usuario funcional a través de la frontera y hacia un proceso funcional del que la entrada forma parte. Si la entrada a un proceso funcional comprende más de un grupo de datos, cada uno describiendo un objeto de interés diferente, se identifica una

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<p>entrada por cada grupo de datos que entra. (Véase también la sección 3.5.7 sobre 'Movimientos de datos únicos y posibles excepciones')</p> <p>b) Una Entrada no deberá sacar datos a través de la frontera, ni leer ni escribir datos del/hacia el almacenamiento persistente.</p> <p>Reglas</p> <p>a) El grupo de datos de una Entrada desencadenante puede constar de sólo un atributo de datos que simplemente informa al software que 'un evento Y ha ocurrido'. Muy a menudo, especialmente en el software de negocio, el grupo de datos de una Entrada desencadenante tiene varios atributos que informan al software que 'un evento Y ha ocurrido y aquí están los datos sobre ese evento en particular'.</p> <p>b) Las marcas de tiempo (ticks) de reloj que están desencadenando eventos serán siempre externos al software que está siendo medido. Por lo tanto, por ejemplo, una marca de tiempo (tick) de reloj cada 3 segundos estará asociada a una Entrada que mueve un grupo de datos con un sólo atributo. Se debe notar que no hay ninguna diferencia si el evento desencadenante es generado periódicamente por el hardware o por otra parte del software fuera de los límites del software medido.</p> <p>c) Salvo que sea necesario en un proceso funcional específico, obtener el tiempo desde el reloj del sistema, este no será considerado como causa de una Entrada.</p> <p>d) Si una ocurrencia de un evento específico desencadena la Entrada de un grupo de datos que comprende hasta "n" atributos de un objeto de interés en particular y los FUR permiten que otras ocurrencias del mismo evento puedan desencadenar una Entrada de un grupo de datos que tiene valores de atributos de sólo un subconjunto de 'n' atributos del objeto de interés, entonces será identificada una entrada, compuesto por todos 'n' atributos.</p> <p>e) Cuando se identifican las Entradas en una pantalla que permite a los usuarios funcionales humanos introducir los datos de entrada en los procesos funcionales, se debe analizar sólo las pantallas que están llenos de datos. Ignore cualquier pantalla con formato pero de otra manera 'en blanco' a excepción de posibles valores por defecto, y no hacer caso de todos los campos y otros encabezados que permiten a los usuarios humanos comprender los datos de entrada requeridos.</p> <p>NOTA. Puede que sea necesario tener en cuenta los campos y otros encabezados en la medición de cambios de los FUR en las Entradas - ver sección 4.4.1.</p>
3.5.3	<p>Salidas (Exit, X)</p> <p>Principios</p> <p>a) Una Salida deberá mover un único grupo de datos describiendo un solo objeto de interés desde el proceso funcional del que la salida forma parte a través de la frontera hacia un usuario funcional. Si la salida de un proceso funcional comprende más de un grupo de datos, hay que identificar una salida para cada grupo de datos que sale. (Véase también la sección 3.5.7 sobre 'Movimientos de datos únicos y posibles excepciones'.)</p> <p>b) Una Salida no introducirá datos a través de la frontera, ni leer ni escribir datos del/hacia el almacenamiento persistente.</p> <p>Reglas</p> <p>a) Una consulta que emite el texto fijo, (donde 'fijo' significa que el mensaje no contiene valores de datos variables, por ejemplo, el resultado de pulsar un botón de 'Términos y Condiciones' en un sitio web comercial), se modela como que tiene una sola Salida por la salida de texto fijo.</p>

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<p>NOTA: Para la salida de la funcionalidad de 'Ayuda', consulte la 'Guía para el dimensionamiento de Aplicaciones de Software de Negocio'. Para la salida de los mensajes relacionados con las condiciones de error o confirmación de éxito, véase la sección 3.5.11 de este Manual de Medición.</p> <p>b) Si la Salida de un proceso funcional mueve un grupo de datos que comprende hasta 'n' atributos de datos de un objeto particular de interés y los FUR permiten que el proceso funcional pueda tener una ocurrencia de una Salida que mueve un grupo de datos que tiene valores para un sub-conjunto de solamente 'n' atributos del objeto de interés, entonces se identifica una Salida, que comprende todos los 'n' atributos de datos.</p> <p>c) Cuando se hace la identificación de Salidas, ignorar todos los campos y encabezados que permiten a los usuarios humanos comprender los datos de salida.</p> <p>NOTA: Puede que sea necesario tener en cuenta el campo y otros encabezados en la medición de cambios de los FUR en Salidas - véase la sección 4.4.1</p>
3.5.4	<p>Lectura (Read, R)</p> <p>Principios</p> <p>a) Una Lectura deberá mover un único grupo de datos describiendo un solo objeto de interés del almacén persistente a un proceso funcional del cual la lectura forma parte. Si el proceso funcional debe recuperar más de un grupo de datos del almacén, se debe identificar una Lectura para cada grupo de datos que es recuperado. (Véase también la sección 3.5.7 sobre los 'Movimientos de datos únicos y posibles excepciones'.)</p> <p>b) Una Lectura no recibirá ni sacará datos a través de la frontera ni escribirá datos al almacenamiento persistente.</p> <p>c) Durante un proceso funcional, el movimiento o manipulación de constantes o variables que son internas del proceso funcional y que sólo se pueden cambiar por un programador, o mediante los resultados intermedios en un cálculo, o de los datos almacenados en un proceso funcional sólo resultantes de la aplicación, más que de los FUR, no se considerará como un movimiento de datos de Lectura.</p> <p>d) Una Lectura siempre incluye cualquier funcionalidad de 'solicitud de lectura' (así, un movimiento de datos separado nunca será contado para cualquier funcionalidad de 'solicitud de lectura'). Véase también la sección 3.5.9.</p> <p>Reglas</p> <p>a) Identifica una Lectura cuando, según los FUR, el software que se está midiendo debe recuperar un grupo de datos desde el almacenamiento persistente.</p> <p>b) No identificar una Lectura cuando los FUR del software que se está midiendo especifican ALgún usuario funcional de software o hardware como la fuente de un grupo de datos, o como los medios de recuperación de un grupo de datos almacenados. (Para este caso ver los principios y reglas para las Entradas y Salidas.)</p>
3.5.5	<p>Escritura (Write, W)</p> <p>Principios</p> <p>a. Una Escritura deberá mover un único grupo de datos describiendo un solo objeto de interés del proceso funcional del que la Escritura forma parte hacia el almacén persistente. Si el proceso funcional debe pasar más de un grupo de datos al almacén persistente, identificar una Escritura por cada grupo de datos que se mueve al almacén persistente. (Véase también la sección 3.5.7 sobre 'Movimientos de datos únicos y posibles excepciones').</p> <p>b. Una escritura no recibirá ni sacará datos de la frontera, ni leerá los datos.</p>

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<p>c. Un requisito para borrar un grupo de datos de un almacén persistente se medirá como una sola Escritura.</p> <p>d. Lo siguiente no podrá considerarse como movimientos de datos de Escritura:</p> <ul style="list-style-type: none"> • El movimiento o manipulación de los datos que no existía en el inicio de un proceso funcional y que no se ha hecho persistente cuando el proceso funcional es completado; • Creación o actualización de variables o resultados intermedios que son internos al proceso funcional; • Almacenamiento de los datos por un proceso funcional resultante sólo de la aplicación, en lugar de que estén establecidos en los FUR. (Un ejemplo podría ser el almacenamiento de datos en forma temporal durante un largo proceso de ordenamiento en un trabajo de procesamiento por lotes.) <p>Reglas</p> <p>a. Identificar una Escritura en que, de acuerdo a los FUR, el software que se está midiendo debe mover un grupo de datos hacia el almacenamiento persistente.</p> <p>b. No identificar una Escritura cuando los FUR del software que se está midiendo especifica algún usuario funcional de software o hardware como el destino del grupo de datos o como medio de almacenamiento del grupo de datos. (Para ver este caso los principios y reglas para las Entradas y Salidas.)</p>
3.5.6	<p>Manipulaciones de datos asociadas con los movimientos de datos</p> <p>Principio</p> <p>Todas las manipulaciones de datos en un proceso funcional serán asociadas con los cuatro tipos de movimiento de datos (E, X, R, y W). Por convención, los movimientos de datos de un proceso funcional se supone que también representan la manipulación de los datos del proceso funcional.</p> <p>Reglas</p> <p>a. Un movimiento de datos de Entrada representa toda manipulación de datos para permitir que un grupo de datos sea introducido por un usuario funcional (por ejemplo, de formato y manipulaciones de presentación) y ser validado,</p> <p>b. Un movimiento de datos de Salida representa toda manipulación de datos para crear los atributos de datos de un grupo de datos que va a ser la salida y/o para permitir que el grupo de datos sea la salida (por ejemplo, de formato y manipulaciones de presentación) y para ser transferido al usuario funcional destinado,</p> <p>c. Un movimiento de datos de Lectura representa todo cálculo y/o procesamiento lógico necesario para recuperar un grupo de datos desde el almacenamiento persistente.</p> <p>d. Un movimiento de datos de Escritura representa todo cálculo y/o procesamiento lógico para crear o actualizar un grupo de datos que se escribe, o eliminar un grupo de datos.</p> <p>e. La manipulación de datos asociada con cualquiera de estos movimientos de datos no incluye ningún tipo de manipulación de datos que se necesita después de que el movimiento de datos se ha completado con éxito, ni tampoco incluye ningún tipo de manipulación de datos asociada con cualquier otro movimiento de datos.</p>
3.5.7	<p>Movimientos de Datos Únicos y posibles excepciones Reglas</p> <p>Nótese que todas las reglas de COSMIC que se refieren a tipos de usuarios funcionales, grupos de datos, movimientos de datos, procesos funcionales y objetos de interés. Para facilitar la lectura, normalmente omitimos 'tipo' de estos términos. Pero En la regla d) a continuación, incluimos 'tipo' donde es útil distinguir un 'tipo' de un 'acontecimiento'.</p>

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<p>a) A menos que los Requisitos Funcionales de Usuario sean dados como en las reglas o c), todos los datos que describen cualquier objeto de interés que se requiere para ser introducido en un proceso funcional deberá ser identificado como un grupo de datos movido por una Entrada.</p> <p>NOTA: Un proceso funcional puede, por supuesto, tener varias entradas, cada una moviendo datos que describen un objeto de interés diferente.</p> <p>La misma regla equivalente se aplica a cualquier movimiento de datos de Lectura, Escritura o de Salida en cualquier proceso funcional.</p> <p>b) Un Requisito Funcional de Usuario puede especificar diferentes grupos de datos que deben introducirse en un proceso funcional por parte de los usuarios funcionales que deben ser identificados por separado para ese proceso funcional, donde cada grupo de datos describe el mismo objeto de interés. Una entrada se identificará para cada uno de estos diferentes grupos de datos.</p> <p>La misma regla equivalente se aplica para las Salidas de datos a diferentes usuarios funcionales de cualquier proceso funcional.</p> <p>NOTA: Cualquier proceso funcional debe tener sólo una Entrada desencadenante.</p> <p>c) Un Requisito Funcional de Usuario puede especificar diferentes grupos de datos a ser movidos desde el almacenamiento persistente a un proceso funcional, cada uno describiendo el mismo objeto de interés. Una Lectura se identificará para cada uno de estos diferentes grupos de datos. La misma regla equivalente se aplica para el Escrituras en cualquier proceso funcional dado.</p> <p>d) No se contarán las ocurrencias repetidas de cualquier tipo de movimiento de datos cuando se está ejecutando. El número de ocurrencias cuando se ejecuta software son irrelevantes para la medida del tamaño funcional.</p> <p>Esto se aplica incluso si múltiples ocurrencias del tipo de movimiento de datos difieren en su ejecución debido a diferentes valores de los atributos de datos del grupo de datos movido resultando en diferentes rutas de procesamiento siendo seguido por el tipo de proceso funcional.</p>
3.5.9	<p>Un proceso funcional requiere datos de un usuario funcional</p> <p>Reglas</p> <p>a) Un proceso funcional deberá obtener un grupo de datos por medio de un movimiento de datos de Entrada de un usuario funcional, <i>cuando el proceso funcional no tiene que decirle al usuario qué datos va a enviar</i>, como en cualquiera de los siguientes cuatro casos:</p> <ul style="list-style-type: none"> • Cuando un usuario funcional envía un grupo de datos a través de una Entrada desencadenante que inicia del proceso funcional; • Cuando un proceso funcional, habiendo recibido un grupo de datos a través de una Entrada desencadenante, espera, expectante el próximo grupo de datos a través de una Entrada desde el usuario funcional (típica en el software de aplicación de negocio, de un usuario funcional humano). • Cuando un proceso funcional, habiéndose iniciado, pide al usuario funcional 'envíame tus datos ahora, si tienes alguno', y el usuario funcional envía sus datos. • Cuando un proceso funcional, habiéndose iniciado, inspecciona los datos de un usuario funcional y recupera los datos que necesita. <p>En estos dos últimos casos (típico de un sistema de votación en tiempo real), por convenio, no habrá una Salida desde el proceso funcional que deba ser identificada para obtener los datos requeridos. El proceso funcional sólo debe enviar un mensaje a un usuario funcional para ingresar sus datos y la funcionalidad del mensaje se considera parte de la entrada. El usuario funcional sabe lo que se puede enviar y el</p>

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<p>proceso funcional sabe lo que puede esperar. Sólo una entrada es necesaria para este caso.</p> <p>b) Cuando un proceso funcional debe obtener los servicios de un usuario funcional (por ejemplo, para obtener datos) y <i>las necesidades de los usuarios le digan lo que debe enviar</i> (normalmente cuando el usuario funcional es otra parte del software fuera del alcance del software que se mide), un movimiento de datos de Salida seguida de uno de Entrada serán identificados. La Salida contiene la solicitud específica de los datos; la Entrada contiene los datos.</p>
3.5.10	<p>Comandos de control en el dominio de aplicaciones de negocio</p> <p>Regla</p> <p>En una aplicación con interfaz para humanos los 'comandos de control' serán ignorados, ya que no implican ningún movimiento de datos sobre un objeto de interés.</p>
3.5.11	<p>Mensajes de Error/Confirmación y otras indicaciones de condiciones de error</p> <p>Reglas</p> <p>a) Una salida se identificará para contabilizar todos los tipos de mensajes de error/confirmación emitidos por un proceso funcional del software que se mide por todas las causas posibles de acuerdo a su FUR, por ejemplo, éxitos o fracasos de: validación de los datos introducidos o para una llamada para recuperar datos o para hacerlos persistentes, o para la respuesta de un servicio solicitado a otra pieza de software.</p> <p>NOTA: Si los FUR del proceso funcional no requieren ningún tipo de mensaje de error/confirmación a emitir, no identificar ninguna Salida correspondiente.</p> <p>b) Si un mensaje a un usuario funcional humana proporciona datos además de confirmar que los datos introducidos han sido aceptados, o que los datos introducidos es por error, entonces estos datos adicionales deben ser identificados como un grupo de datos movido por una Salida, además de la Salida de error/confirmación.</p> <p>c) Todos los demás datos, emitidos o recibidos por el software que se mide, hacia/desde su hardware o usuarios funcionales software deben ser analizados de acuerdo con los FUR como Salidas o Entradas, respectivamente, de acuerdo con las reglas normales COSMIC, independientemente de si o no los valores de los datos indican una condición de error.</p> <p>d) Las Lecturas y Escrituras se consideran que incluyen cualquier informe de condiciones de error. Por lo tanto ninguna Entrada al proceso funcional que se mide se debe identificar para cualquier indicación de error recibido como resultado de una Lectura o Escritura de datos persistentes.</p> <p>e) Ninguna Entrada o Salida se identificarán para cualquier mensaje que indica una condición de error que pueda ser emitido, mientras se utiliza el software que se está midiendo, pero que no se requiere que sea procesado en modo alguno por el FUR de ese software, por ejemplo, un mensaje de error emitido por el sistema operativo.</p>
4.3.1	<p>Agregando los resultados de la medición</p> <p>Reglas</p> <p>a. Para cualquier proceso funcional, el tamaño funcional de cada movimiento de datos individual debe ser agregado en un único valor de tamaño funcional en unidades de CFP para luego sumar todos juntos.</p> $\text{Tamaño (proceso funcional)} = \sum \text{tamaño(Entradas}_i) + \sum \text{tamaño(Salidas}_i) + \sum \text{tamaño(Lecturas}_i) + \sum \text{tamaño(Escrituras}_i)$

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<p>b. Para cualquier proceso funcional, el tamaño funcional de los cambios en sus Requisitos Funcionales de Usuario se sumarán al tamaño de movimientos de datos que han sido añadidos, modificados o eliminados en el proceso funcional para dar un tamaño del cambio en unidades de CFP, según la siguiente fórmula.</p> $\text{Tamaño (Cambio(proceso funcional))} = \sum \text{tamaño (movimientos datos añadidos)}_i + \sum \text{tamaño (movimientos datos modificados)}_i + \sum \text{tamaño (movimientos datos eliminados)}_i$ <p>Para más información sobre el agregado del tamaño funcional, véase la sección 4.3.2. Para medir el tamaño de los cambios en el software, véase la sección 4.4.2.</p> <p>c. El tamaño de una pieza de software dentro de alcance definido se obtendrá sumando los tamaños de sus procesos funcionales, sujeto a las normas e) y f) a continuación.</p> <p>d. El tamaño de cualquier cambio en una pieza de software dentro de un alcance definido se obtendrá sumando los tamaños de todos los cambios de todos los procesos funcionales, sujeto a las normas e) y f) a continuación.</p> <p>e. Los Tamaños de piezas de software o de cambios de piezas de software podrán sumarse sólo si se mide al mismo nivel de granularidad de los FUR el proceso funcional.</p> <p>f. Los Tamaños de piezas de software y/o cambios en los tamaños de piezas de software dentro de una capa o de diferentes capas serán sumados sólo si tiene sentido hacerlo, a efectos de la medición.</p> <p>g. El tamaño de una pieza de software se obtiene sumando los tamaños de sus componentes (independientemente de cómo se descompone) proporcionados</p> <ul style="list-style-type: none"> a. el tamaño de las contribuciones de los movimientos de datos inter-componentes son eliminadas y b. sólo una Salida se identifica para todos los mensajes de error/confirmación emitidos por un proceso funcional a un usuario funcional humano. <p>h. Si el método COSMIC se extiende localmente (por ejemplo para medir algún aspecto del tamaño no cubierto por el método estándar), entonces el tamaño medido por la extensión local debe ser informado por separado como se describe en la sección 5.1 y no puede ser añadido al tamaño obtenido por el método estándar, medido en CFP (véase además la sección 4.5)</p>
4.4.1	<p>Modificando un movimiento de datos</p> <p>Reglas</p> <ul style="list-style-type: none"> a. Si un movimiento de datos debe ser modificado debido a un cambio de la manipulación de los datos relacionados con el movimiento de datos y/o debido a un cambio en el número o tipo de los atributos del grupo de datos movido, un cambio CFP se medirá, independientemente de la cantidad real de modificaciones en el movimiento de la datos. b. Si un grupo de datos debe ser modificado, los movimientos de datos moviendo el grupo de datos modificado, cuya funcionalidad no se ve afectada por la modificación del grupo de datos, no serán identificados como movimientos de datos modificados. <p>NOTA: Una modificación de cualquier dato que figuren en entrada o salida por pantalla que no esté relacionado con el objeto de interés para un usuario funcional no será identificados como un cambio CFP (ver sección 3.3.4 para los ejemplos de estos datos.)</p>
5.1	<p>Etiquetado de las mediciones COSMIC</p> <p>Reglas</p> <p>Los resultados de una medición COSMIC se deberían anotar como 'x CFP(v.y)', donde:</p>

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<ul style="list-style-type: none"> • 'x' representa el valor numérico del tamaño funcional • 'v.y' representa la identificación de la versión estándar del método COSMIC usado para obtener el valor numérico del tamaño funcional "x". <p>Nota: Si un método de aproximación local se usó para obtener la medida, pero aparte la medida se hizo utilizando las normas de la versión estándar COSMIC, la regla de etiquetado anterior debería utilizarse, pero en algún sitio se deberá anotar el uso del método de aproximación.-ver sección 5.2.</p> <p>Etiquetado de extensiones locales COSMIC</p> <p>Regla</p> <p>Un resultado de una medición COSMIC utilizando extensiones locales debería ser anotado como:</p> <p style="text-align: center;">' x CFP (v. y) + z Local FP', donde:</p> <ul style="list-style-type: none"> • 'x' representa el valor numérico obtenido al agregar todos los resultados de medidas individuales de acuerdo al método estándar COSMIC, versión v.y. • 'v.y' representa la identificación de la versión estándar del método COSMIC usado para obtener el valor numérico de tamaño funcional "x". • 'z' representa el valor numérico obtenido al agregar todos los resultados individuales de medida obtenidos de las extensiones locales al método COSMIC.
5.2	<p>Reportes de mediciones COSMIC</p> <p>Regla</p> <p>Además de las medidas reales, descritas en el punto 5.1, todos o algunos de los siguientes atributos de cada medición se debe registrar, dependiendo del propósito de medición y el nivel deseado de comparabilidad con otras medidas, por ejemplo, para fines de evaluación comparativa.</p> <ol style="list-style-type: none"> Identificación del componente de software medido (nombre, versión ID o Configuración ID) Las fuentes de información usadas para identificar el FUR utilizado para la medición. El ámbito del software. Una descripción de la arquitectura en capas en las que la medición es realizada, si aplica. Una especificación del propósito de la medición. Una descripción del alcance de la medición y su relación con el alcance global de una serie de medidas relacionadas, si las hubiera. (Usar las categorías de alcance genéricas de la sección 2.2) El patrón de medida utilizado (COSMIC o local), con el modo de procesamiento (online o por lotes). Los usuarios funcionales del software El nivel de granularidad de los artefactos de software disponibles y el nivel de descomposición del software. El punto en el ciclo de vida del proyecto cuando se hizo la medición (especialmente si la medida es una estimación basada en FUR incompletas, o si en realidad se hizo sobre la bases de funcionalidad entregada). El objetivo o margen de error que se considera para la medida

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
	<p>l. Indicaciones de si el método de medición estándar COSMIC fue utilizado, y/o se usó una aproximación local al método estándar y/o se utilizaron extensiones locales (ver sección 4.5). Usar las reglas de etiquetado de las secciones 5.1 y 5.2.</p> <p>m. Una indicación de si la medida es de funcionalidad desarrollada o entregada (funcionalidad ‘desarrollada’ se obtiene creando nuevo software; funcionalidad ‘entregada’ incluye funcionalidad desarrollada y también incluye funcionalidad obtenida por otros medios diferentes a crear nuevo software, p.ej: incluir todas las formas de reutilización de software existente, el uso de parámetros existentes para añadir o cambiar funcionalidad, etc.).</p> <p>n. Una indicación de si la medida es de funcionalidad recientemente proporcionada o si es el resultado de una actividad de ‘mejora’ (p.ej: la suma es de funcionalidad añadida, cambiada o borrada-ver 4.4)</p> <p>o. El número de componentes importantes, si es aplicable, cuyos tamaños han sido añadidos para el tamaño total registrado.</p> <p>p. El porcentaje de funcionalidad implementada por el software de re-utilización.</p> <p>q. Para cada alcance dentro del alcance global de la medición, una matriz de medida, como se especifica en el apéndice A</p> <p>r. El nombre del medidor y cualquier certificación de COSMIC, la fecha de la medición.</p>

Apéndice E

APÉNDICE E – LOS PRINCIPALES CAMBIOS DE VERSIÓN 3.0.1 A LAS VERSIONES 4.0 Y V4.0.1

Este apéndice contiene un resumen de los principales cambios introducidos en la evolución del método de medición de tamaño funcional COSMIC partir de la versión 3.0.1 a la versión 4.0 y respecto de la versión actual v4.0.1.

Para seguir la evolución anterior del método, por favor consulte el Manual de Medición para cada versión anterior del método (2.2, 3.0 y 3.0.1).

A 'MUB' es un Boletín de Actualización del Método, publicado entre las versiones principales del Manual de Medición para anunciar y explicar cambios propuestos.

E1: Principales cambios respecto de v3.0.1 a v4.0

V4.0 Ref	Cambio
-	Simplificación de numeración de figuras aplicada, secuenciado por capítulos, en lugar de por sección (subsección) (por ejemplo, "Figura 3.5.8.3 'ahora' Figura 3.8 ').
Caps.1 a 5	Un resumen se ha añadido para cada uno de los capítulos 1 - 5.
1.1	Las secciones 1.1.2, 1.1.3 y 1.1.4 sobre 'limitaciones del método' se han trasladado a la sección 4.5. El título de 1.1.1 se ha eliminado ya que su texto es ahora el de la sección completa 1,1.
1.2	La definición de 'Requisitos Funcionales de Usuario' se ha movido de la sección 2.2 y es más calificado cuando se utiliza en el método COSMIC (MUB 11).
1.2.3	Una nueva sección 1.2.3 sobre 'Requisitos No funcionales' se ha agregado (MUB 10).
1.3.1	El principio G) del Modelo Contextual de Software de COSMIC (SCM) fue simplificado (MUB 7) y posteriormente se ha eliminado debido a fusionarse con el principio a) Del Modelo Genérico Software (GSM) donde pertenece de manera más lógica.
1.5	Esta nueva sección se refiere a 4.5, que ahora contiene las tres 'limitaciones' discutido previamente en las secciones 1.1.2, 1.1.3 y 1.1.4
2.2	Las reglas para el 'Alcance' ahora son las reglas para una 'Medición del Alcance'.
2.2.2	La definición de 'Capa' se ha revisado para que sea consistente con los conceptos SEI (MUB 9). La Figura 2.4 se ha añadido para mostrar como las Capas dependen de la arquitectura del software.
2.2.2	La definición y principios de 'componentes semejantes' se han eliminado ya que no son necesarios en el método COSMIC (MUB 9)
2.3.2	La Nota 1 a la definición de almacenamiento persistente ha sido rephraseado para mayor claridad (MUB 7) Esta definición estaba en la sección 3.3.1 en la MM v3.0.1.
2.3.3	Una sección sobre 'Diagramas de Contexto' se ha añadido.

2.4.3	El gran ejemplo de los niveles de granularidad del sistema 'Everest' se ha reescrito para mayor claridad
3.1	El principio 'Aplicando el Modelo Genérico de Software' se ha eliminado en virtud de que ya está en la descripción del SCM y de GSM, y en la descripción del proceso de medición.
3.2	Las definiciones de los conceptos de eventos desencadenantes, usuarios funcionales, entradas desencadenantes, y procesos funcionales se han mejorado (MUB 11).

V4.0 Ref	Cambio
3.2	Los posibles grados de las relaciones (i.e. 'cardinalidades') entre los eventos, usuarios funcionales, entrada desencadenantes y procesos funcionales se especifican de forma más completa, ilustrados en la figura 3.3, y algunos ejemplos se dan en el Apéndice C (MUB 11).
3.2.2	El proceso de identificación de procesos funcionales esta ahora descrito con más claridad
3.2.2	Las Reglas para un proceso funcional se han actualizado y simplificado (MUB 11).
3.2.3 b)	La discusión de como analizar y medir aplicaciones de negocio con proceso batch se han ampliado y hecho mucho más claras, agregando la nueva Figura. 3.4 (MUB 11)
3.2.7	Una nueva sección se ha añadido para explicar cómo medir procesos funcionales que comparten alguna funcionalidad en común o que son muy similares (MUB 11)
3.2.8	Una nueva sección se ha añadido para explicar la necesidad de distinguir entre el evento desencadenante que empieza en un sistema de software, y evento desencadenante que empieza en un proceso funcional (MUB 11).
3.5	La antigua sección 4.1 'Identificación de los movimientos de datos' se ha trasladado al capítulo 3, sección 3.5.
3.5.1	La Figura 3.5 (Antigua Fig. 4.1.1) muestra más claramente la relación entre un proceso funcional y los cuatro tipos de movimientos de datos (E, X, R and W).
3.5.5	El Principio d) para una escritura ha sido modificado para evitar una contradicción con el principio c) (MUB 7).
3.5.6	Las declaraciones sobre la manipulación de los datos asociados a los movimientos de datos son de hecho Reglas y se han proporcionado con un cuadro de Reglas. El fraseo se ha simplificado.
3.5.6 (and 4.4)	Notas al pie No. 13 y 14 estaban equivocadas. Se han corregido.
3.5.7	La Regla b) la cual se refiere a cuando dos o más movimientos de datos del mismo tipo mueven datos sobre el mismo objeto de interés pueden identificarse en el mismo proceso funcional. Se ha ampliado en dos Reglas b y c), dejándolo más claro. Varios ejemplos se han agregado en esta sección.
3.5.8	Varios cambios se han hecho en esta sección. El Ejemplo 1 se ha dividido en dos ejemplos (1 y 4). El Ejemplo 3 se ha agregado e ilustra el caso de cuando el software que se está midiendo tiene permitido o no acceder a los datos requeridos en el almacenamiento persistente (MUB 7 v2). El análisis para los mensajes de error ahora se discute para cada

	uno de estos ejemplos, en referencia a la nueva sección 3.5.11 de las reglas sobre los mensajes de error/ confirmación.
3.5.10	El concepto de 'comando de control' se ha generalizado a cualquiera aplicación con una interfaz humana.
3.5.11	Una nueva sección se ha agregado con una definición y reglas precisas para la medición de los mensajes de error/confirmación y otras condiciones de error.
4.4.1	La Nota 1 de las Reglas de Modificación de un movimiento de datos se han eliminado ya que son inconsistentes: Un cambio de valor de código no es un cambio de un tipo de atributo (por ejemplo. No se debe de contar). Pero un cambio requerido de formato o formato de encabezado que incluso no es un cambio de un tipo de atributo, debe de ser contado.
4.4.1	La definición de los cambios de un movimiento de datos, y por lo tanto al grupo de datos o para la manipulación de datos asociado se ha convertido en una definición de 'modificación (de la funcionalidad de un movimiento de datos)'.
4.5	Las Secciones 1.1.2, 1.1.3 y 1.1.4 sobre las limitaciones del método COSMIC se han movido para convertirse en las secciones 4.5.2, 4.5.3 y 4.5.4 respectivamente. En la sección anterior 1.1.2 (ahora 4.5.2) 'Dominio No-aplicable' se ha re-titulado como 'Manipulación de software rico en datos' y el primer párrafo de esta sección ha sido sustituido, como la practica ha demostrado que el método COSMIC puede ser utilizado para medir el tamaño de varios tipos de software que figuran listados en este establecimiento de 'limitaciones (MUB 8).
V4.0 Ref	Cambio
Referencias	Todas las referencias están ahora en esta sección.
Apéndice A	El diagrama se ha actualizado para mostrar los totales.
Apéndice B	Un Apéndice se ha agregado con ejemplos de requisitos No funcionales que se desarrollan en parte o totalmente en requisitos funcionales de usuario (MUB 10)
Apéndice C	Un Apéndice se ha añadido con ejemplos y diagramas de la cardinalidad de las relaciones entre eventos desencadenantes, usuarios funcionales, entradas desencadenantes y procesos funcionales.
Apéndice D	Resumen de principios y normas se ha actualizado.
Apéndice F	El Glosario de términos ha pasado de 'Documentación y Glosario de Términos' documento (V3.0) a este Apéndice F de MM v4.0, y ha sido actualizado.

E2: Principales cambios de v4.0 a v4.0.1

V4.0.1 Ref	Cambio
Prefacio	El documento de introducción debe leerse primero si usted es un principiante o está convirtiendo, esto se ha enfatizado mediante la adición de un título al párrafo actual.
1.2.3	La definición de un "Requerimiento No Funcional" se ha hecho más clara y ahora no incluyen los requisitos y restricciones del proyecto. Figura 1.3 se ha cambiado en consecuencia.

1.3.3	Una nueva sección 1.3.3 'Tipos contra Ocurrencias' se ha agregado. COSMIC nunca había definido "tipo" y "ocurrencia" de forma explícita y parecía que algunas veces los medidores tenían dificultad para comprender la diferencia.
3.2.1	La definición de 'proceso funcional' fue ambigua en un punto ya que no era claro si 'único' se refería a 'el conjunto de movimiento de datos' o a 'la parte elemental de FUR'. Se ha resuelto mediante la adición de una coma después de 'movimientos' y el cambio de dos ocurrencias de 'Ese' a 'estos'.
3.3.1	La definición de un 'grupo de datos' se ha simplificado ya que era innecesariamente compleja. Dos de los principios se han eliminado ya que no agregan valor.
3.3.1	La definición de un 'objeto de interés' se ha revisado y se ha añadido una nota, ambas con el objetivo de hacer la definición más clara.
3.5.1	La definición de un 'movimiento de datos', 'Entrada', 'Salida', 'Lectura' y 'Escritura' Ahora afirman que cada uno se considera 'cuenta' (no incluye) cualquiera manipulación de datos relacionada.
3.5.6	El texto que ahora hace hincapié que las Reglas de esta sección solo son relevantes para la medición de los cambios en los FUR existentes. Las Reglas que definen qué tipo de manipulaciones de datos se asocian con cuales movimientos de datos ahora dejan claro que la manipulación de datos 'se asocia con', no está 'incluida' en los movimientos de datos.
3.5.7	Las Reglas para 'Movimiento de datos únicos y posibles excepciones' se han reformulado ya que eran complejas. Un error se ha corregido (dos movimientos de datos no pueden ser considerados como diferentes debido a que tienen diferente FUR para su manipulación de datos asociada estaría en contradicción con el principio ya que toda manipulación de datos en un proceso funcional debe ser asociado con los cuatro tipos de movimientos de datos (E, X, R y W)). Se han añadido dos ejemplos y han sido secuencializados nuevamente para alinearse con las Reglas revisadas. También se ha hecho algunas simplificaciones editoriales.
V4.0.1 Ref	Cambio
3.5.11	La definición de 'mensaje de error/confirmación' se ha clarificado. Parte de la definición anterior se ha hecho una regla.
4.3.1	La Regla g) de cómo obtener el tamaño de una pieza de software a partir del tamaño de sus componentes se ha cambiado de una sentencia 'doblemente-negativa' a una positiva. La regla se ha extendido sobre la forma de agregar mensajes de error/confirmación.
Varios	Las referencias a las normas ISO/IEC 14143/1 sobre la definición de conceptos (de métodos de FSM) y de ISO/IEC 19761 las cuales definen los principales conceptos del método COSMIC eran confusas y ahora se han corregido.
Glosario	Las definiciones de 'entrada' y 'salida' se han simplificado ya que eran innecesariamente complejas.

APÉNDICE F – GLOSARIO DE TÉRMINOS

Los siguientes términos se utilizan a través del método de medición de tamaño funcional COSMIC (el 'Método COSMIC'), de acuerdo con las definiciones que se encuentran en esta sección. Términos ya definidos por la ISO, como 'Medición de Tamaño funcional' o 'unidad de medida', junto con su definición ISO también se han adoptado para el método COSMIC.

Para muchos de los términos que figuran en el glosario, en su caso, se muestra el sufijo 'tipo'. Dado que cualquier método de medición de tamaño funcional tiene como objetivo identificar 'tipos' y no 'ocurrencias' de datos o funciones, casi invariablemente a lo largo del método COSMIC vamos a preocuparnos por 'tipos' y no 'ocurrencias'. En consecuencia, en los textos vamos a caer con el sufijo 'tipo' de estos términos por el bien de la legibilidad, excepto cuando específicamente necesitamos distinguir el tipo y la ocurrencia. Esta es también la convención adoptada en el Estándar Internacional (ISO / IEC 19761: 2011) definición del método COSMIC. De vez en cuando esta convención conduce a dificultades en la elaboración de estas definiciones - véase por ejemplo la Nota 3 de la definición de 'Tipo de movimiento de datos' a continuación, que no aparece en el Estándar Internacional.

Para una discusión más completa de "tipo" versus "ocurrencia", véase la sección 1.3.3.

NOTA: Los términos que se utilizan sólo en las 'guías' COSMIC de un dominio específico se definen en estas guías; no se muestran a continuación. En las definiciones que se indican a continuación:

- términos que se definen en otra parte de este glosario están subrayados, para facilitar la referencia cruzada.
- términos que se originan en la norma ISO para el método COSMIC (ISO / IEC 19761) o que son de otra manera específica al método COSMIC se muestran en **negrita y cursiva**.
- otros términos que han sido adoptados del estándar ISO, pero que no son específicos al método COSMIC se muestran en **negritas**.

Ajuste (de una medida). El proceso de convertir una medida del tamaño en una unidad de medición, a una medida en otra unidad de medición.

Alcance (de una medición). El conjunto de los Requisitos funcionales de usuario a ser incluido en una instancia específica de medición de tamaño funcional. [17]

NOTA: (Específico para el método COSMIC.) Una distinción debe hacerse entre el 'alcance general', es decir, todo el software que se debe medir de acuerdo con el propósito, y el 'alcance' de cualquier pieza individual de software en el alcance global, cuyo tamaño se debe medir por separado. En el Manual de Medición, el término 'alcance' (o la expresión 'alcance de la medición') se referirán a una pieza individual de software cuyo tamaño debe medirse por separado.

Almacenamiento Persistente. Almacenamiento que permite a un proceso funcional almacenar datos más allá de la vida del proceso funcional y/o que permite a un proceso funcional recuperar datos almacenados por otro proceso funcional, o almacenados por una ocurrencia previa del mismo proceso funcional o almacenados por algún otro proceso.

NOTA 1: En el modelo COSMIC, el almacenamiento persistente es un concepto que solo existe dentro de los límites del software que se está midiendo, no puede por lo tanto, ser considerado como un usuario funcional del software que se está midiendo.

NOTA 2: Un ejemplo de 'algún otro proceso' sería en la creación de memoria de solo lectura.

Ambiente operativo (software). El conjunto de software que opera concurrentemente sobre un sistema informático específico del que la aplicación de software depende.

Aplicación. Un sistema de software para la recolección, guardado, procesamiento y la presentación de datos por medio de un ordenador.

NOTA: Esta es una adaptación de la definición que figura en la Norma ISO/IEC 24570:2005 ingeniería de software -- NESMA método de medición de tamaño funcional versión 2.1.

(Definición alternativa de ‘aplicación de software’). Software diseñado para ayudar a los usuarios a realizar tareas particulares o para manejar determinados tipos de problemas, a diferencia del software que controla la propia computadora.

NOTA: Esta es una ligera adaptación de la definición que figura en la Norma ISO/IEC 24765:2010 Sistemas y software de ingeniería-vocabulario, 4.5).

Atributo de Datos - Tipo Atributo de Datos (sinónimo ‘tipo elemento de datos’). La menor unidad de información, identificada dentro de un tipo de grupo de datos, que lleva un significado desde la perspectiva de los Requisitos Funcionales de Usuario del software.

Capa. Una partición funcional de arquitectura de un sistema de software.

Comando de Control. Un comando que permite a los usuarios funcionales humanos controlar el uso del software pero que no implica ningún movimiento de datos sobre un objeto de interés de los FUR del software que se está midiendo.

NOTA: Un comando de control no es un movimiento de datos porque el comando no mueve datos sobre un objeto de interés. Ejemplos de comandos son ‘page up/down’ Presionar Tab o Enter, hacer clic en el “OK” para confirmar una acción anterior, presionar un botón para continuar, etc.

Componente. Cualquier parte de un sistema de software que está separado por razones de arquitectura de software, y/o que se ha especificado, diseñado, o desarrollado por separado.

Componente Funcional Base (BFC). Una *unidad elemental de los Requisitos Funcionales de Usuario* definidos por un método FSM para fines de medición [17].

NOTA: El método COSMIC define un tipo de movimiento de datos como un BFC.

Componente Funcional Base– Tipo Componente Funcional Base - (tipo BFC). Una categoría definida de BFC’s [17]. El método COSMIC tiene cuatro tipos de BFC, Entradas, Salidas, Lecturas y Escrituras (-tipos).

Datos de Aplicación-general. Cualquier dato relacionado con la aplicación en general y no relacionado con un objeto de interés de un proceso funcional específico.

Datos de Entrada (Input). Los datos movidos por todas las Entradas de un proceso funcional.

Datos de Salida (Output). Los datos movidos por todas las Salidas de un proceso funcional.

E. Abreviatura de “tipo de Entrada”.

Entrada - Tipo Entrada. Un movimiento de datos que mueve un grupo de datos de un usuario funcional a través de la frontera hacia el proceso funcional donde se requieren.

NOTA: Un tipo entrada se considera que incluye cierta manipulación de datos asociada – ver el manual de medición para más detalles.

Entrada Desencadenante – Tipo Entrada Desencadenante. El movimiento de datos de Entrada de un proceso funcional que mueve un grupo de datos, generado por un usuario funcional, que el proceso funcional necesita para inicial el procesamiento.

NOTA: Los FUR para un proceso funcional pueden requerir una o más Entradas además de la entrada desencadenante.

Escritura – Tipo Escritura. Un movimiento de datos que mueve un grupo de datos que se encuentra dentro de un proceso funcional al almacenamiento persistente.

Nota: una Escritura se considera que incluye una cierta manipulación de datos asociada – ver manual de medición para más detalles.

Evento - Tipo Evento. Algo que sucede.

Evento Desencadenante – Tipo Evento Desencadenante. Un evento reconocido en los Requisitos Funcionales de Usuario del software que se está midiendo, genera que uno o más usuarios funcionales del software que se está midiendo generen uno o más grupo de datos, cada uno de los cuales posteriormente serán movidos por una entrada desencadenante. Un evento desencadenante no puede ser sub-dividido.

NOTA: Los eventos de reloj o temporización pueden ser eventos desencadenantes.

Frontera. Una interface conceptual entre el software que se está midiendo y sus usuarios funcionales.

NOTA: Se deduce de la definición que hay una frontera entre dos piezas de software en las misma o diferentes capas que intercambian datos donde una pieza de software es un usuario funcional de la otra, y /o viceversa.

Grupo de Datos – Tipo Grupo de Datos. Un conjunto de atributos de datos distinto, no-vacío y no-ordenado donde cada tipo de atributo de datos incluido describe un aspecto complementario del mismo objeto de interés.

Lectura - Tipo Lectura. Un movimiento de datos que mueve un grupo de datos de un almacenamiento persistente dentro de un proceso funcional que lo requiere.

NOTA: Un tipo lectura es considerado para cierta manipulación de datos asociada – Consulte el manual de medición para más detalles.

Manipulación de datos. Cualquier cosa que le ocurre a los datos que no sea un movimiento de datos dentro o fuera de un proceso funcional, o entre un proceso funcional y almacenamiento persistente.

Medición del Tamaño Funcional (FSM). El proceso de medir del tamaño funcional. [17]

Mensaje de Error/confirmación. Una Salida emitida por un proceso funcional a un usuario humano que, o bien confirma solamente que los datos introducidos han sido aceptados, o solo que hay un error en los datos introducidos.

NOTA: Cualquier Salida que puede incluir indicaciones de error, pero que no está destinado a un usuario funcional, no es un mensaje de error/confirmación.

Método de Medición [14]. Una secuencia lógica de operaciones, que se describe de forma genérica, utilizada para la realización de mediciones.

Método de Medición del Tamaño Funcional. *Una aplicación específica del* FSM definida por un conjunto de REGLAS, que se ajusta a las características obligatorias de la norma ISO/IEC 14143-1:1998. [17]

Modelo [16]. Una descripción o analogía utilizada para ayudar a visualizar un concepto que no se puede observar directamente.

Modificación (de la funcionalidad de un movimiento de datos)

a) Un movimiento de datos se considera que se modifica funcionalmente si al menos una de las siguientes situaciones aplica:

- El grupo de datos movido se modifica
- La manipulación de datos asociada se modifica

b) Un grupo de datos se modifica si al menos una de las siguientes situaciones aplica:

- Uno o más atributos nuevos se agregan al grupo de datos
- Uno o más atributos existentes se eliminan del grupo de datos
- Uno o más atributos existentes se modifican, por ejemplo: en el significado o en el formato (pero no en sus valores)

c) Una manipulación de datos se modifica si su funcionalidad cambió de alguna manera.

Movimiento de Datos – Tipo Movimiento de datos. Un componente funcional base que mueve un solo tipo de grupo de datos.

NOTA 1: Hay cuatro sub-tipos del tipo de movimiento de datos, llamados: (tipos) Entrada, Salida, Lectura y Escritura.

NOTA 2: Para fines de medición, cada movimiento de datos se considera que contiene cierta manipulación de datos asociada – para más detalles consulte el Manual de medición.

NOTA 3: Mas precisamente, es una ocurrencia de un movimiento de datos, no un tipo de movimiento de datos, que realmente *mueve* las ocurrencias del grupo de datos (no tipos). Este comentario también se aplica a las definiciones de Entrada, Salida, Lectura y Escritura.

Nivel de descomposición. Cualquier nivel que resulte de dividir una pieza de software en componentes.

(Llamado “nivel 1”, por ejemplo), y luego de dividir los componentes en sub-componentes (‘Nivel 2’), luego de dividir sub-componentes en sub-sub componentes (Nivel 3’), etc.

NOTA 1: No se debe confundir con ‘nivel de granularidad’.

NOTA 2: Las mediciones de tamaño de los componentes de una pieza de software solo pueden ser directamente comparables para los componentes en el mismo nivel de descomposición.

Nivel de granularidad. Cualquier nivel de expansión de la descripción de una sola pieza de software (por ejemplo. Una declaración de sus requisitos, o una descripción de la estructura de la pieza de software) de tal manera que en cada nivel de expansión incrementado, la descripción de la funcionalidad de la pieza de software está en un nivel más elevado y uniforme de detalle.

NOTA: Los medidores deben ser conscientes que cuando los requisitos están evolucionando en la vida de un proyecto de software, en cualquier momento se han documentado diferentes partes de la funcionalidad del software requerido en distintos niveles de granularidad.

Objeto de Interés - Tipo Objeto de Interés. Cualquier ‘cosa’ en el mundo del usuario funcional que se identifica en los Requisitos Funcionales del Usuario sobre el software que se requiere para procesar datos. Puede ser cualquier cosa física, así como cualquier objeto conceptual o parte de un objeto conceptual.

NOTA 1: En la definición anterior, ‘procesar’ puede significar cualquier operación para mover y/o manipular dos datos.

NOTA 2: En el método COSMIC, el término ‘objeto de interés’ se utiliza con el fin de evitar los términos relacionados con métodos de ingeniería de software específicos. El termino no implica ‘objetos’ en el sentido utilizado en métodos Orientados a Objetos.

Patrón de Medición (Estrategia). Una plantilla estándar que se puede aplicar en la medición de una pieza de software desde un dominio funcional de software, que define los tipos de usuario funcional que pueden interactuar con el software, el nivel de descomposición del software y los tipos de movimientos de datos que el software puede manejar.

Pieza de software. Cualquier elemento de software en cualquier nivel de descomposición desde el nivel de un sistema de software completo hacia abajo e incluyendo el nivel del componente más pequeño de un sistema de software.

Piezas semejantes de software. Dos piezas de software son semejantes entre sí en caso que residan en la misma capa.

Proceso Funcional - Nivel de Granularidad. Un Nivel de granularidad de la descripción de una pieza del software en el cual:

Los usuarios funcionales son seres humanos individuales o dispositivos de ingeniería o piezas de software (y no cualquier grupo de estos) y

Eventos individuales ocurren tal que la pieza de software debe responder (y no cualquier nivel de granularidad en el que se definen grupos de eventos). Ver nota 3 abajo.

NOTA 1: En la práctica, la documentación del software y los requisitos funcionales del usuario, por lo tanto a menudo describen la funcionalidad en diferentes niveles de granularidad, sobre todo cuando la documentación está todavía en evolución.

NOTA 2: “Los grupos de estos” (usuarios funcionales) podrían, por ejemplo, ser un ‘departamento’ cuyo miembros pueden manejar muchos tipos de procesos funcionales, o un ‘panel de control’ que tiene muchos tipos de instrumentos, o “sistemas centrales”.

NOTA 3: ‘Los grupos de eventos’ podrían, por ejemplo, ser indicados en un comunicado del FUR en un alto nivel de granularidad por un flujo de entrada a un sistema de software de contabilidad etiquetado ‘Operaciones de Venta’ o por un flujo de entrada a un sistema de software etiquetada ‘pilot commands’.

Proceso Funcional - Tipo Proceso Funcional

Un conjunto de movimientos de datos, el cual representa una parte elemental de los requisitos funcionales del usuario para el software que se está midiendo, que es único dentro de estos FUR y que puede ser definido independientemente de cualquier otro proceso funcional en estos FUR

Un proceso funcional puede tener solo un evento desencadenante. Cada proceso funcional inicia el procesamiento al recibir un grupo de datos movido por el movimiento de datos entrada desencadenante del proceso funcional.

El conjunto de todos los movimientos de datos de un proceso funcional es el conjunto que se necesita para cumplir con los FUR para todas las posibles respuestas a la entrada desencadenante.

NOTA 1: Cuando se implementa, es una ocurrencia de un proceso funcional que inicia la *ejecución* con la recepción de una *ocurrencia* de un grupo de datos movido por una *ocurrencia* de una entrada desencadenante.

NOTA 2: El FUR para un proceso funcional puede requerir una o más Entradas además de la entrada desencadenante.

NOTA 3: Si un usuario funcional envía un grupo de datos con error. Por ejemplo: porque un sensor de usuario está funcionando mal o por una orden mal introducida por un usuario humano, por lo general es la tarea del proceso funcional determinar si realmente el evento ocurrió y/o si los datos introducidos son realmente válidos. Y cómo responder.

Propósito de una medición. Declaración que define por qué se está realizando una medición, y para que se utilizará el resultado

R. Abreviatura de 'tipo Lectura'.

Requisitos Funcionales de Usuario (FUR). Un sub-conjunto de las necesidades de los usuarios. Requisitos que describen lo que el software debe hacer, en términos de tareas y servicios.

NOTA 1: Requisitos Funcionales de Usuario se refieren a, pero no se limitan a:

- Transferencia de datos (por ejemplo datos de clientes de entrada, enviar señal de control);
- Transformación de datos (por ejemplo Calcular los intereses bancarios, Derivar la temperatura media);
- almacenamiento de datos (por ejemplo almacenar pedidos de clientes,
- registrar la temperatura ambiente a través del tiempo); recuperación de datos (por ejemplo listar los empleados actuales, Recuperar posición de una aeronave).

Ejemplo de necesidades de los usuarios que son los Requisito de usuario no Funcionales incluyen, pero no se limitan a (aunque algunos de ellos pueden llegar a ser verdaderos FUR por el momento la solución está completamente definida):

- Limitaciones de calidad (por ejemplo la facilidad de uso, fiabilidad, eficiencia y portabilidad);
- Limitaciones organizativas (por ejemplo lugares de operación, el hardware y el cumplimiento de las normas de destino);
- Limitaciones ambientales (por ejemplo interoperabilidad, la seguridad y privacidad);
- Limitaciones de aplicación (por ejemplo lenguaje de desarrollo, calendario de entrega).

Requisito No-Funcional. Cualquier requisito para la parte de software de un sistema hardware/software o un producto de software, incluyendo la forma en que debe ser desarrollado y mantenido, y como se debe llevar a cabo en la operación, salvo un requisito funcional para el software. Requisitos No-funcionales se refieren a:

- La calidad de software;
- El medio ambiente en el que el software debe aplicarse y que debe servir.
- Los procesos y la tecnología que se utilizará para desarrollar y mantener el software;
- La tecnología que se utiliza para la ejecución del software.

NOTA: Requisitos del sistema o software que inicialmente se expresan como No-funcionales frecuentemente evolucionan conforme un proyecto avanza total o parcialmente en FUR para el software.

Salida - Tipo Salida. Un Movimiento de datos que mueve un grupo de datos de un proceso funcional a través de la frontera hacia el usuario funcional que lo requiera.

NOTA: Un tipo Salida se considera para reportar cierta manipulación de datos asociada – consulte el Manual de medición para más detalles.

Sistema. *Una combinación de hardware, software y procedimientos manuales organizados para lograr los propósitos establecidos.*

NOTA: La definición anterior es una adaptación de ISO/IEC 15288:2008. En la definición COSMIC, 'hardware, software y "procedimientos manuales"' sustituye 'elementos que interactúan' en ISO/IEC definition.

Sistema de Software. Un sistema que se compone solo de software.

Software [17]. Un conjunto de instrucciones de ordenador, datos, procedimientos y tal vez la documentación que opera como un todo, para cumplir con un conjunto específico de objetivos, todos los cuales se puede describir desde una perspectiva funcional a través de un conjunto finito de Requisitos funcionales de usuario, requisitos técnicos y de calidad.

Sub proceso - Tipo Sub proceso. Una parte de un proceso funcional que o bien mueve los datos (dentro del software de un usuario funcional o fuera del software de un usuario funcional, o hacia o desde el almacenamiento persistente) o que manipula datos.

Tamaño Funcional. Un tamaño del software derivado de la cuantificación de los Requisitos Funcionales de Usuario. [17]

Unidad de medida [14]. Una cantidad determinada, definida y aprobada por la convención, con la que se comparan otras cantidades de la misma naturaleza con el fin de expresar sus magnitudes relativas a esa cantidad. Es de señalar que las unidades de medida han asignado convencionalmente nombres y símbolos.

Ver también 'Unidad de medida COSMIC'

Unidad de medida COSMIC. 1 CFP (Punto Funcional Cosmic, Cosmic Function Point), que se define como el tamaño de un movimiento de datos.

NOTA: La unidad de medida se conoce como un 'Cfsu' (COSMIC functional size unit) antes de la v3.0 del método.

Usuario [17]. Cualquier persona o cosa que se comunica o interactúa con el software en cualquier momento.

NOTA: Ejemplos de 'cosa' incluyen, pero no se limitan a, aplicaciones de software, animales, sensores, u otro hardware.

Usuario Funcional. Un tipo de usuario que es un emisor y/o un destinatario de los datos en los requisitos funcionales de usuario de una pieza de software.

Valor (de una cantidad) [14]. La magnitud de una cantidad determinada, generalmente expresado como una unidad de medida multiplicado por un número.

Abreviatura de "tipo Escritura".

Abreviatura para 'tipo Salida'.

APÉNDICE G - CAMBIO DE SOLICITUD Y PROCEDIMIENTO DE COMENTARIO

El Comité de Prácticas de Medición COSMIC (MPC) está deseoso de recibir retroalimentación y comentarios, y si necesario, solicitudes de cambio de esta guía. En este Apéndice se establece la forma de comunicarse con el COSMIC MPC. Todas las comunicaciones al COSMIC MPC deben ser enviadas por e-mail a la siguiente dirección: mpc-chair@cosmic-sizing.org

Comentarios generales y retroalimentación

Comentarios y/o retroalimentación relativas a la guía, tales como las dificultades de la comprensión o de la aplicación del método COSMIC, sugerencias para la mejora general, etc deben ser , enviados por e-mail a la dirección antes mencionada. Los mensajes serán registrados y confirmados generalmente dentro de las dos semanas siguientes a la recepción. El MPC no puede garantizar la acción de los comentarios generales.

Solicitudes de cambio formales

Cuando el lector de la guía considere que hay un defecto en el texto, una necesidad de aclaración, o que algún texto necesita mejorar, una solicitud de cambio formal ('CR') podrá ser presentado. Las CR's se registran y se confirman dentro de las dos semanas siguientes a la recepción. Cada CR se le asigna un número de serie el cual se distribuirá a los miembros de COSMIC MPC, un grupo mundial de expertos del método COSMIC. Su ciclo normal de revisión toma como mínimo un mes pudiendo tomar más tiempo si la CR resulta difícil de resolver. El resultado de la revisión puede ser que el CR pueda ser aceptada o rechazada, o en espera de nuevos debates (en el último caso, por ejemplo si hay una dependencia de otra CR), y el resultado se comunicará al solicitante lo antes posible.

Un CR formal será aceptado solo si se documenta con toda la siguiente información.

Nombre, cargo y organización de la persona que presenta la CR.

Datos de contacto de la persona que presenta la CR.

Fecha de presentación

Establecimiento general de la finalidad de la CR (por ejemplo 'la necesidad de mejorar el texto...').

Texto real que se necesita cambiar, reemplazar o eliminar (o clara referencia al mismo).

Propuesta adicional o texto de reemplazo.

Explicación completa de por qué es necesario el cambio.

Un formulario para la presentación de un CR está disponible en el portal de www.cosmic-sizing.org . La decisión de COSMIC MPC sobre los resultados de una revisión de CR y, de ser aceptada, en cual versión se aplicará la CR, es final.

Preguntas sobre la aplicación de método COSMIC

COSMIC MPC lamenta que no es capaz de responder a las preguntas relacionadas con el uso o la aplicación del método COSMIC. Existen organizaciones comerciales que pueden ofrecer capacitación y asesoría o soporte de herramientas para el método. Por favor consulte el sitio www.cosmic-sizing.org para más detalles.