El Método de Medición de Tamaño Funcional COSMIC

Manual de Medición

(La Guía de implementación de COSMIC para ISO/IEC 19761: 2017)

Versión 3.5.3.0.2 diciembre 2017

Agradecimientos

Revisores de la versión 4.0.2		
Diana Baklizky TI Métricas Brasil	Jean-Marc Desharnais Escuela Superior de Tecnología – Universidad de Québec Canada	Peter Fagg Pentad Reino Unido
Cigmed Gencel Universidad Libre de Bozen-Bolzano	Arlan Lesterhuis* Países Bajos	Dylan Ren Measures Technology LLC
Bruce Reynolds Investigaciones Tecolote Estados Unidos	Hassan Soubra ESTACA Francia	Charles Symons* Reino Unido
Francisco Valdés Souto Spingere Mexico	Frank Vogelezang Metri Países Bajos.	Chris Woodward Reino Unido

* Editores de la versión 4.0.2 del método de COSMIC

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

Equipo de traducción de la Versión		
Francisco Valdés Souto SPINGERE Mexican Software Metrics Association (AMMS) National Autonomous University of Mexico - Science Faculty México	Ricardo Cárdenas Estrada Coppel México	

Se puede encontrar una versión de dominio público del Manual de medición COSMIC y otros informes técnicos, incluidas las traducciones a otros idiomas, en la Base de conocimientos de www.cosmic-sizing.org.

Control de Versiones

La siguiente tabla muestra el historial de las versiones de este documento

FECHA	REVISOR (ES)	Modificación/Adición
1999-03-31	Serge Oligny	Primer borrador, emitido para comentarios a revisores
1999-10-29	Equipo Principal COSMIC	Revisado, comentarios finales antes de publicar “prueba de campo”. Versión 2.0
2001-05-01	Equipo Principal COSMIC	Revisado para estar en conformidad con ISO/IEC 14143-1: 1998 + aclaraciones sobre las reglas de medición a la versión 2.1
2003-01-31	Comité de Prácticas de Medición COSMIC	Revisado para estar en conformidad con ISO/IEC FDIS19761:2002 + aclaraciones adicionales sobre las reglas de medición a la versión2.2
2007-09-01	Comité de Prácticas de Medición COSMIC	Revisado para obtener más aclaraciones y adiciones a las reglas de medición a la versión 3.0, particularmente en el área de la fase de la estrategia de medición. El nombre del método se cambió del “método COSMIC-FFP” al “método COSMIC”. Al actualizar a V3.0 desde V2.2 en otros documentos.
2009-05-01	Comité de Prácticas de Medición COSMIC	La versión 3.0 se revisó a v3.0.1 para realizar mejoras editoriales menores y aclaraciones, y para distinguir los ejemplos con mayor claridad. Esta versión también incorpora los cambios propuestos en los boletines de actualización de métodos 3, 4 y 5.
2014-04-01	Comité de Prácticas de Medición COSMIC	Versión 4.0 revisada para tener en cuenta los Boletines de actualización de métodos del 6 al 11, varias mejoras editoriales e incorporar el Glosario de términos. Vea el Apéndice E para detalles de estos cambios.
2015-04-01	Comité de Prácticas de Medición COSMIC	Versión 4.0.1 revisada para tener en cuenta un error en v4.0 y varias otras mejoras editoriales. Vea el Apéndice E para detalles de estos cambios.
Octubre 20017	Comité de Prácticas de Medición COSMIC	Versión 4.0.2 revisada para tener en cuenta los MUB 12, 13 y 14 y varias mejoras editoriales. Vea el Apéndice E para detalles de estos cambios.
Diciembre 20017	Comité de Prácticas de Medición COSMIC	La versión de octubre de 4.0.2 ha sido corregida por algunos errores editoriales menores, especialmente en el Glosario. Los cambios al Glosario también se describen en el Apéndice E.

Prefacio

El método COSMIC ofrece un método estandarizado para medir un tamaño funcional del software de los dominios comúnmente conocidos como software de “aplicaciones de negocios” (o “MIS”), software de “tiempo-real”, software de “infraestructura” y algunos tipos de software científico/ingeniería.

El método COSMIC fue aceptado originalmente por ISO/IEC JTC1 SC7 como una norma internacional en diciembre del 2002. La versión actual es ISO/IEC 1976:2017 “Ingeniería de Software -COSMIC- Un método para la medición de tamaño funcional [1] (en lo sucesivo, “ISO/IEC 19761”).

El estándar ISO/IEC 19761 contiene solo las definiciones normativas y las reglas del método como en la versión 4.0 del método. El propósito del Manual de Medición es proporcionar estas reglas y definiciones, y también proporcionar una explicación y muchos más ejemplos en orden para ayudar a los medidores a comprender totalmente como aplicar el método. El Manual de Medición es la principal descripción estándar del método COSMIC para uso práctico.

Introducción al método COSMIC

Además del “Manual de Medición”, la introducción al método de medición de software COSMIC” [2] ofrece un resumen del método. Este documento de “Introducción” debe ser leído primero por cualquiera que sea nuevo en la medición de tamaño funcional (“FSM”, por sus siglas en inglés) o que esté familiarizado con otro método FSM y esté pensando convertirlo, o que simplemente quiera una descripción general del método COSMIC, antes de leer este manual. Mucha información antecedente sobre FSM y el método COSMIC, así como las Guías de apoyo (ejemplo, sobre cómo aplicar el método en diversas circunstancias especiales, etc.), casos de estudios, trabajos de investigación, etc., se pueden encontrar en el portal de www.comsmic-sizing.org.

Principales cambios en la versión 4.0.2 del método COSMIC

Esta v4.0.2 no introduce cambios en los principios básicos de método. Los únicos cambios que afectan las definiciones y reglas; y todos están diseñados para mejorar la compresión y la repetibilidad de las mediciones. Los cambios más importantes ocurrieron en las secciones 3.2 y 3.3, y se originan a partir de los boletines de actualización de métodos (MUB) que ya se han publicado, de la siguiente manera.

- Se modificó la definición de “Evento Desencadenante” para aclarar que solo el primer grupo de datos generado por un usuario será movido por la “Entrada Desencadenante”. La posibilidad de un “rol dual” del usuario funcional (también puede ser un objeto de interés) se hizo más claro en la definición de usuario funcional (MUB 12).
- Una nueva regla para “identificar diferentes grupos de datos (y por lo tanto diferentes objetos de interés) movido en el mismo proceso funcional han sido agregados (MUB 13)[1].
- En la definición de “objeto de interés” la frase “proceso y/o movimiento de datos” ha sido reemplazado por “movimiento del grupo de datos hacia adentro o fuera del software, hacia o desde el almacenamiento persistente”. Estos cambios tienen como objetivo de aclarar que un objeto de interés se puede identificar solo si es el sujeto o un grupo de datos es movido. Un objeto de interés no debe identificarse como un sujeto de manipulación de datos solamente (MUB14)

Ver Apéndice E para obtener una lista detallada de todos los cambios realizados desde la versión 4.0 a v4.0.1 y desde v4.0.1 a v4.0.2. Este Manual de medición para la versión 4.0.2 del método se convierte en la definición estándar actual del método desde su fecha de publicación.

Consecuencias de los cambios principales a v4.0.2 en medidas de tamaño existentes, etc.

Los principios básicos originales del método COSMIC se han mantenido sin cambios desde que se publicaron por primera vez en el primer borrador del Manual de Medición en 1999. Esto es a pesar de los diversos refinamientos y adiciones necesarios para introducir el Estándar Internacional y para producir todas las versiones del método hasta esta última versión 4.0.2.

Los tamaños funcionales medidos de acuerdo con los principios y las reglas de la versión 4.0.2 del Manual de Medición pueden diferir de los tamaños medidos usando versiones anteriores solo porque las nuevas reglas

[1] Estas nuevas reglas se publicaron por primera vez en la Guía para medir el software de aplicaciones de negocios, versión 1.3, publicada en mayo de 2017.

pretenden ser más precisas y completas. Por lo tanto, los Medidores tienen menos discreción para la interpretación personal de las reglas de lo que era posible con versiones anteriores.

Como una indicación adicional de la continuidad del método, cualquier persona que haya aprobado el examen de certificación a nivel Fundamentos para la versión 3.0/3.0.1/4.0 del método se considerará que aún está certificado para las versiones v4.0.1 y v4.0.2 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.

El Capítulo 1 trata los tipos de software para los cuales se puede usar el método COSMIC. El término “Requisitos funcionales del usuario” (“FUR”) se define, junto con los principios básicos del método COSMIC. También se definen el proceso de medición del método COSMIC y la unidad de medida.

El Capítulo 2 describe la "Estrategia de medición", la primera fase del proceso para medir una pieza de software, en términos de parámetros clave como el propósito de la medición, el alcance de la medición y los usuarios funcionales del software. Estos parámetros deben definirse antes de comenzar a medir para que el significado de las mediciones resultantes pueda acordarse y entenderse.

El Capítulo 3 trata la segunda fase "Mapeo" del proceso de medición al definir cómo se debe asignar el FUR a los procesos funcionales y sus movimientos de datos. El resultado de esta fase será el "modelo" COSMIC del FUR que se puede medir.

El capítulo 4 describe la fase final "Medición" del proceso de medición. Define reglas para asignar un tamaño al FUR de una pieza de software y cómo agregar tamaños de diferentes piezas de software. Este capítulo también trata cómo dimensionar los cambios en el software y discute la posibilidad de "extensiones locales" al método estándar COSMIC.

El Capítulo 5 enumera los parámetros que deben considerarse para registrar la medición.

Los apéndices de la A a la E proporcionan más detalles, un resumen de los principios y reglas del método, y los cambios principales en este Manual de medición de 4.0 a v4.0.1 y al presente 4.0.2. El 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 en métricas de software de todo el mundo, fundada en 1998. Todas sus publicaciones son "abiertas" y están disponibles para su distribución gratuita, sujeto a restricciones de derechos de autor y reconocimiento. Para obtener más información sobre COSMIC y su organización, consulte el sitio web de COSMIC www.cosmic-sizing.org.

El Comité de Prácticas de Medición COSMIC

Tabla de Contenido

1

INTRODUCCION

1.0 Resumen del capitulo

Este capítulo tiene cuatro propósitos:

- Explicar los tipos de software (“dominios aplicables”) para los que se pueden utilizar el método COSMIC, y las limitaciones de su uso.
- Para definir ‘Requisitos funcionales del usuario’ (‘FUR’), es decir, los requisitos para la funcionalidad del software que el método COSMIC pretende medir. Explicamos en términos generales cómo un Medidor puede extraer o derivar el FUR de los artefactos de software disponibles para una medición de tamaño funcional. Los Requisitos No Funcionales ("NFR") también se definen, ya que los requisitos que inicialmente se expresan como no funcionales a menudo evolucionan a medida que un proyecto avanza parcial o totalmente a FUR que también se puede medir.
- Definir los principios básicos del método COSMIC, como se resumen en dos modelos. El "Modelo Contextual de software" se utiliza para caracterizar una pieza de software para medir. El ‘Modelo Genérico de Software’ define los principios clave del modelo COSMIC de la 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 medida 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 dominios:

- Software de aplicación de negocios que normalmente se necesita en apoyo de la administración de empresas, como banca, seguros, contabilidad, personal, compras, distribución o fabricación, etc. Este software se caracteriza a menudo como "rico en datos", ya que está dominado en gran medida por la necesidad de gestionar grandes cantidades de datos sobre eventos y "cosas" en el mundo real.
- Software en tiempo real, cuya tarea es mantenerse al día o controlar los eventos que ocurren en el mundo real. Algunos ejemplos serían software para centrales telefónicas y conmutación de mensajes, software integrado en dispositivos para controlar máquinas como electrodomésticos, elevadores, vehículos y aeronaves, para el control de procesos y la adquisición automática de datos, cualquier software utilizado en el "Internet de las cosas" y software Dentro del sistema operativo de las computadoras.
- Software de infraestructura compatible con los anteriores, como componentes reutilizables, controladores de dispositivos y similares.
- Algunos tipos de software científico/de ingeniería.

1.2. Requisitos Funcionales del Usuario

El método COSMIC consiste en la aplicación de un conjunto de modelos, principios, reglas y procedimientos que miden los Requisitos Funcionales del Usuario (FUR) para una pieza de software dada. El resultado es un “valor de una cantidad” numérica (como se define por ISO) representando el tamaño funcional de una pieza de software acorde con el método COSMIC.

Los Requisitos Funcionales del Usuario se definen por ISO de la siguiente manera:

DEFINICIÓN – Requisitos Funcionales del Usuario (FUR).

Un subconjunto de requisitos del usuario. Requisitos que describen que es lo que el software debe hacer, en términos de tareas y servicios.

NOTA 1: Los Requisitos Funcionales del Usuario se relacionan, pero no se limitan a:

- transferencia de datos: (por ejemplo, ingresa datos del cliente, enviar una señal de control)
- transformación de datos (por ejemplo, calcular los intereses bancarios, derivar un promedio de temperatura.
- almacenamiento de datos (por ejemplo, almacenar un pedido del cliente, grabar la temperatura durante un periodo)
- recuperación de datos (por ejemplo, lista de empleados vigentes, recuperar la última posición de un vuelo.)

Ejemplo de requisitos del usuario que no son Requisitos Funcionales del Usuario incluyen, pero no están limitados a:

- restricción de calidad (por ejemplo, usabilidad, confiabilidad, eficiencia y portabilidad)
- restricciones de organización (por ejemplo, localización de la operación, hardware, objetivo y cumplimiento de normas)
- restricciones ambientales (por ejemplo, interoperabilidad, seguridad, privacidad y seguridad física.
- Restricciones de aplicación (por ejemplo, lenguaje de programación, calendario de entrega)

NOTA 2: En los documentos de COSMIS el término “FUR” está restringido a significar “El Requisito Funcional del Usuario” que:

- Son derivados de los artefactos disponibles del software (requisitos, diseños, artefactos físicos, etc.)
- Se ajustan, de ser necesario, por supuestos que sobrepasaran la incertidumbre en los artefactos disponibles.
- Contienen toda la información necesaria para la medición de tamaña funcional COSMIC.

De lo contrario usamos la expresión como “requisito funcional”, “requisitos actuales” o “artefactos físicos”, etc., dependiendo del contexto.

Tome en cuenta que el método COSMIC reconoce que algunos tipos de requisitos (ejemplo, calidad y restricciones ambientales) pueden expresarse temprano en la vida del proyecto como “Requisitos NO Funcionales” de acuerdo con la definición ISO. Sin embargo, estos mismos requisitos pueden evolucionar a medida que el proyecto avanza en Requisitos Funcionales del usuario. Ver sección 1.2.3 de este Manual de Medición.

El tamaño funcional medido por el método COSMIC está diseñado para depender solo de los FUR del software a ser medido (incluyendo FUR derivados de NFR) y ser independientes de cualquier requisito o restricciones relativas a la aplicación de los FUR. “Funcionalidad” puede ser vagamente definido como “el procesamiento de la información que el software deben de realizar para sus usuarios.

1.2.1 Extrayendo el Requisito Funcional del Usuario desde los artefactos del software

En el mundo real del desarrollo de software es raro encontrar “artefactos” para el software como los documentos que describen el software en detalle, o manifiestos visibles del software como visualización de pantalla, en la 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 de los artefactos del software disponibles, antes de representarlos como conceptos de los “modelos de software” COSMIC.

Como se ilustra en la Figura 1.1, FUR puede derivarse de artefactos de ingeniería de software que se producen antes de que el software exista. Por lo tanto, el tamaño funcional del software se puede medir antes de su implementación en un sistema informático.

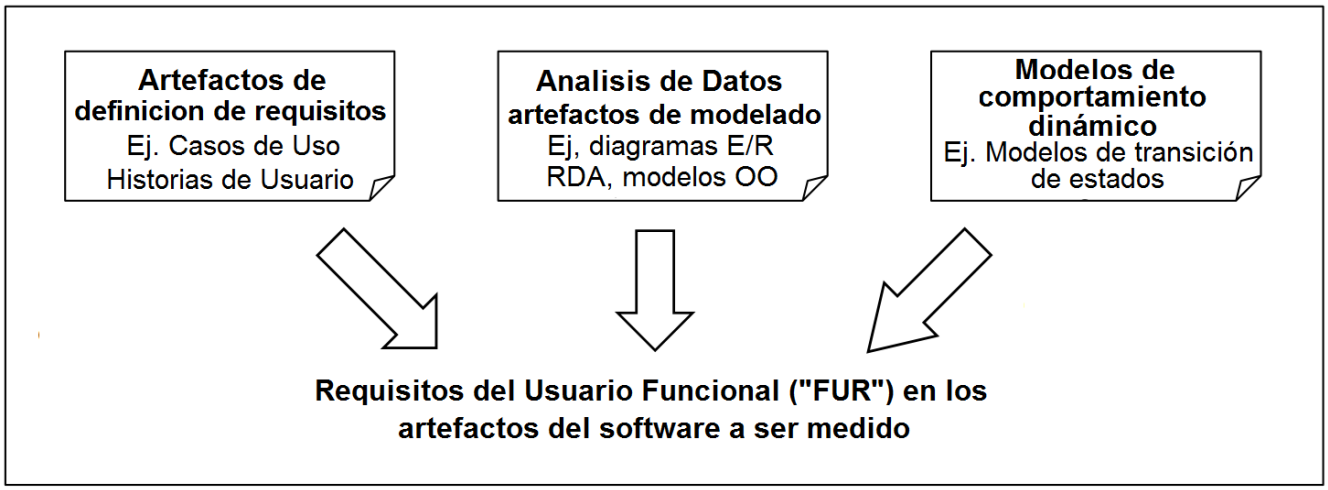


Figura 1.1 - Fuentes previas a la implementación de los requisitos funcionales del usuario

NOTA: Los requisitos pueden producirse incluso antes de que se asignen al hardware o software. Dado que el método COSMIC está dirigido a dimensionar el FUR de una pieza de software, solo se miden los FUR asignados al software. Sin embargo, en principio, COSMIC se puede aplicar a los requisitos antes de que se asignen al software o al hardware, independientemente de la eventual decisión de asignación. Por ejemplo, es sencillo dimensionar la funcionalidad de una calculadora de bolsillo utilizando el método COSMIC sin ningún conocimiento de que hardware o software (si corresponde) está involucrado. Sin embargo, la afirmación de que el método COSMIC se puede usar para dimensionar los requisitos asignados al hardware necesita más pruebas en la práctica antes de que se pueda considerar como completamente validado sin la necesidad de más reglas.

En otras circunstancias, es posible que deba medirse algún software existente sin que haya alguno, o con solo unos pocos, los artefactos de diseño o arquitectura disponibles, y el FUR podría no estar documentado (por ejemplo, para software heredado). En tales circunstancias, todavía es posible derivar el FUR de los artefactos del sistema informático incluso después de que se haya implementado, como se ilustra en la Figura 1.2.

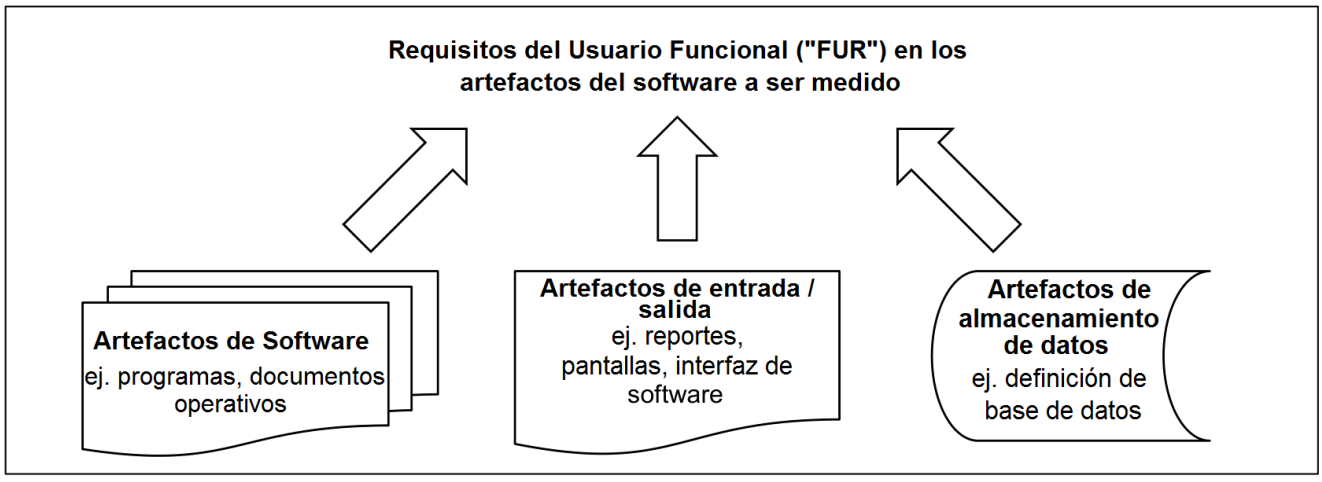


Figura 1.2 - Fuentes posteriores a la implementación de los requisitos de usuario funcional

1.2.2 El proceso de derivar requisitos funcionales de usuario a partir de artefactos de software

El proceso que se utilizará y, por lo tanto, el esfuerzo requerido para extraer el FUR de diferentes tipos de artefactos de ingeniería de software o para derivarlos del software instalado obviamente variará enormemente; Estos procesos no pueden ser tratados en el Manual de Medición. El método supone que los requisitos de usuario funcional del software a medir existen o que pueden extraerse o derivarse de sus artefactos, a la luz del propósito de la medición.

Por lo tanto, el Manual de medición se limita a definir y describir los conceptos de los modelos COSMIC (el 'Modelo Contextual de Software' y el 'Modelo Genérico de Software' - ver la sección 1.3) y cómo aplicarlos para medir el FUR de una determinada pieza de software.[2]

Si el Medidor realmente entiende estos dos modelos, siempre será posible obtener el FUR de una pieza de software que se medirá a partir de sus artefactos disponibles, aunque el Medidor puede tener que hacer algunas suposiciones debido a la información faltante o no clara.

1.2.3 Requisitos no funcionales

La definición estándar de la ISO de los requisitos funcionales del usuario (o "FUR", ver más arriba) enumera varios tipos de "requisitos del usuario" en la Nota 1 que no son FUR. Por implicación, estos son requisitos no funcionales (NFR).

NFR puede ser muy importante para un proyecto de software. En casos extremos, una declaración de requisitos para un sistema de software intensivo puede requerir tanta documentación para el NFR como para el FUR. Pero la distinción entre NFR y FUR no es tan simple como aparece en la definición ISO de FUR. El método COSMIC se puede usar para medir algunos requisitos que pueden expresarse primero como no funcionales. Primero necesitamos definir NFR [18]:

DEFINICIÓN – Requisitos no funcionales (de software)
Cualquier requisito para la parte de software de un sistema de hardware/software o producto de software, incluida la forma en que se debe desarrollar y mantener, y cómo debe funcionar en la operación, excepto un requisito funcional del usuario para el software. Los requisitos no funcionales se refieren a: • la calidad del software; • el entorno en el que se debe implementar el software y al que debe servir; • los procesos y la tecnología que se utilizarán para desarrollar y mantener el software; • La tecnología por utilizar para la ejecución del software. NOTA: los requisitos del sistema o software que inicialmente se expresan como no funcionales a menudo evolucionan a medida que un proyecto avanza total o parcialmente a FUR para el software.

Varios estudios [3] han demostrado que algunos requisitos que inicialmente aparecen como NFR del *sistema* evolucionan a medida que un proyecto avanza hacia una combinación de requisitos que pueden implementarse en funciones de software, y otros requisitos o restricciones que son verdaderamente "no funcionales". Ver figura 1.3. Esto es cierto para muchas limitaciones de calidad del sistema, como el tiempo de respuesta, la facilidad de uso, la capacidad de mantenimiento, etc. Una vez identificadas, estas funciones de software que se han "ocultado" en NFR al comienzo de un proyecto se pueden dimensionar utilizando el método COSMIC al igual que cualquier otra función de software. No reconocer este tamaño funcional "oculto" es una de las razones por las que puede parecer que los tamaños de software aumentan a medida que avanza un proyecto.

[2] Varias Guías de COSMIC, por ejemplo, para dimensionar el software de aplicaciones de negocios [7] y para dimensionar el software en tiempo real [4], se proporciona orientación sobre el mapeo de varios métodos de análisis de datos y determinación de requisitos y de artefactos de software a los conceptos de COSMIC.

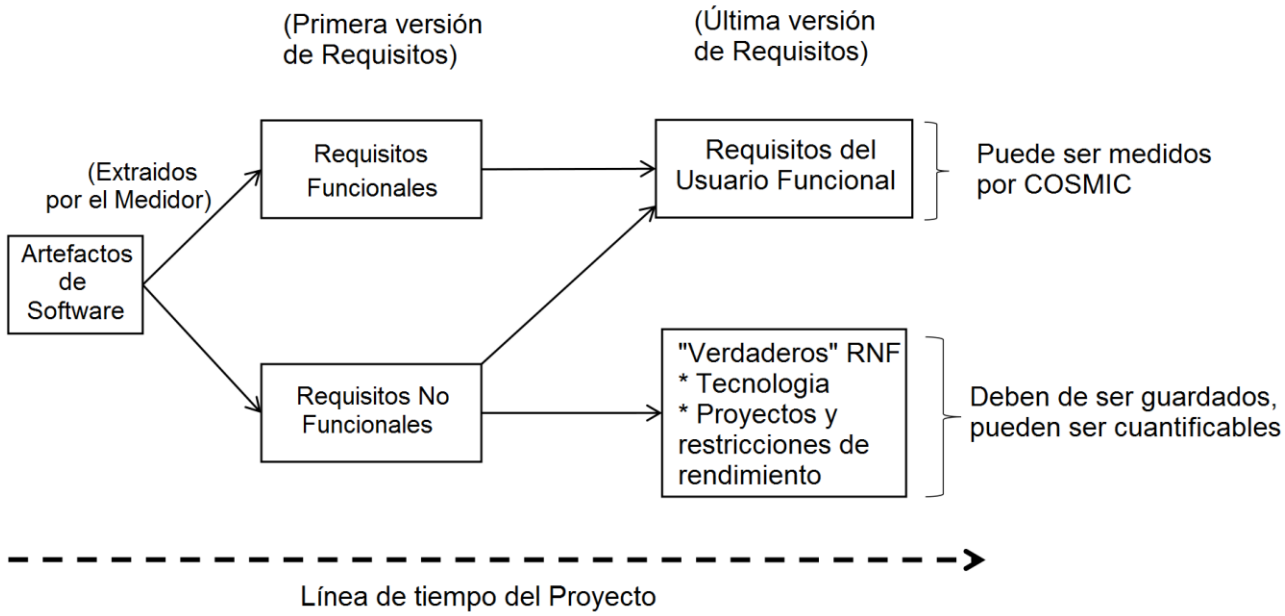


Figura 1.3 - Muchos requisitos que aparecen inicialmente como NFR evolucionan a FUR a medida que avanza un proyecto

EJEMPLO DE NEGOCIO: Los requisitos para un nuevo sistema de software incluyen la declaración "el usuario tendrá la opción de proteger los archivos mediante cifrado". El proyecto para desarrollar el sistema se encuentra en la etapa de estimación de esfuerzo y costo. Se consideran dos opciones:

- *Desarrollar algún software de cifrado propietario. Para fines de estimación del proyecto, puede ser necesario medir el tamaño del FUR para el software de encriptación.*
- *Compre un paquete comercial existente Off-The Shelf (COTS). Para fines de estimación del proyecto, puede ser necesario medir solo el tamaño de la funcionalidad del software necesaria para integrar el paquete COTS. El costo del paquete y el esfuerzo para integrar y probar el paquete de cifrado de archivos también deberán considerarse en la estimación del costo del proyecto.*

EJEMPLO DE TIEMPO REAL: Los requisitos de confiabilidad o tolerancia a fallas para los sistemas aeroespaciales se logran principalmente a través de una combinación de redundancia y respaldo de los sistemas físicos. Una función como la supervisión del motor se implementa en dos o más computadoras integradas separadas. Esta función tiene una restricción de tiempo estricta declarada como NFR: "Cada computadora debe responder dentro de un tiempo específico. Si alguna de las computadoras responde repetidamente después del tiempo requerido, o sus resultados no están de acuerdo con los demás, debe ser rechazado" (por un mecanismo especificado como requisito funcional). Un requisito para la tolerancia a fallas que cuando se indica inicialmente puede aparecer como no funcional, por lo tanto, evoluciona a FUR que se puede medir. El mecanismo de temporización también se puede implementar parcialmente en el software y esta funcionalidad también se puede medir (consulte, por ejemplo, la 'Guía para dimensionar el software en tiempo real' [4], sección 3.2).

Para obtener más ejemplos, consulte el Apéndice B. Todos estos ejemplos demuestran que cuando hay un requisito para medir un tamaño de algún software al principio de la vida de un proyecto, es importante considerar si algún NFR pudiera evolucionar hacia el software FUR y si el tamaño de estos programas FUR también debe ser medido.

1.3 Los principios fundamentales del método COSMIC

El método COSMIC se basa en principios fundamentales 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 quiera medir, el tamaño de una pieza de software se puede medir de muchas maneras, incluso usando la misma unidad de medida. Los principios del 'Modelo Contextual de software' permiten que un Medidor defina el software a medir y la medida del tamaño. Esto garantiza que los futuros usuarios puedan entender e interpretar los resultados de forma coherente.

Los principios del "Modelo genérico de software" definen cómo se modelan los FUR del software a medir para que puedan medirse.

La razón principal para incluir estos dos modelos en esta etapa del Manual de medición es mostrar cómo el método COSMIC es fundamentalmente muy simple. También necesitaremos referirnos a los dos modelos más adelante en el Manual. Sin embargo, un Medidor principiante no debe esperar ser capaz de leer estos dos modelos y luego ir a medir con precisión. Para aplicar los modelos a una situación de medición particular, el Medidor necesitará las definiciones de los diversos conceptos y los principios, reglas, explicaciones y ejemplos adicionales que se dan en este Manual.

Nótese Bien. Los términos que aparecen en negrita cuando se usan por primera vez en las secciones 1.3.1 y 1.3.2 se usan con significados que pueden ser específicos del método COSMIC. Para las definiciones formales, consulte el glosario en el Apéndice F de este Manual de Medición. Las referencias dadas con cada principio son las secciones de este Manual donde se trata el tema en detalle.

1.3.1 El Modelo Contextual de Software COSMIC

PRINCIPIOS – El Modelo Contextual de Software COSMIC.
a) El software está limitado por el hardware
b) El software esta típicamente estructurado en **capas** (2.2.2)
c) Una capa puede tener una o más piezas de software "**similares**" separadas (2.2.2)
d) Cualquier software que deba medirse, se definirá por su **alcance de medición**, que se limitará completamente dentro de una sola capa (2.2).
e) El alcance de una pieza de software a medir dependerá del **propósito** de la medición (2.1).
f) Los **usuarios funcionales** de una pieza de software que se medirá se identificarán a partir de sus **Requisitos de usuario funcional** (FUR) como remitentes y/o destinatarios de datos deseados a/desde el software respectivamente (2.3).
g) Los requisitos funcionales del software pueden expresarse a diferentes **niveles de granularidad** (2.4).
h) Una medida de tamaño COSMIC precisa de una pieza de software requiere que sus FUR sean conocidos en los niveles de granularidad en los que se pueden identificar sus **procesos funcionales** y subprocesos (2.4.3).
i) Si los requisitos funcionales de una pieza de software están disponibles solo en un alto nivel de granularidad, se puede utilizar un enfoque de aproximación para medir un tamaño en el alto nivel de granularidad y para escalar el resultado a un tamaño COSMIC aproximado a los niveles del Procesos funcionales y movimientos de datos. (2.4.3).

1.3.2 El Modelo Genérico de Software.

Habiendo identificado y definido el FUR del software que se medirá en términos del Modelo Contextual del software, ahora aplicamos el Modelo Genérico de Software al FUR para identificar los componentes de la funcionalidad que se medirá. Este Modelo Genérico de Software supone que los siguientes principios generales son válidos para cualquier software que pueda medirse con el método.

PRINCIPIOS – El Modelo Genérico de Software COSMIC.

a) Una pieza de software interactúa con sus usuarios funcionales a través de una **frontera** y con un **almacenamiento persistente** dentro de este límite (2.3).

b) Los requisitos funcionales del usuario de una pieza de software a medir se pueden mapear en procesos funcionales únicos (3.2).

c) Cada proceso funcional consiste en subprocesos (3.2).

d) Un subproceso puede ser un **movimiento de datos** o una manipulación de datos (3.2).

e) Un movimiento de datos mueve un solo **grupo de datos** (3.3).

f) Hay cuatro tipos de movimiento de datos, **Entrada, Salida, Escritura** y **Lectura** (3.5).

- Una Entrada mueve un grupo de datos a un proceso funcional desde un usuario funcional.
- Una Salida saca un grupo de datos de un proceso funcional a un usuario funcional.
- Una Escritura mueve un grupo de datos de un proceso funcional a un almacenamiento persistente.
- Una Lectura mueve un grupo de datos del almacenamiento persistente a un proceso funcional

g) Un grupo de datos consta de un conjunto único de **atributos de datos** que describen un único objeto de interés (3.3).

h) Cada proceso funcional se inicia por su movimiento de datos de **Entrada desencadenante**. El grupo de datos movido por la Entrada desencadenante es generado por un usuario funcional en respuesta a un **evento desencadenante** (3.2).

i) El tamaño de un proceso funcional es igual a la cuenta total de sus movimientos de datos.

j) Un proceso funcional debe incluir al menos el movimiento de datos de Entrada desencadenante y un movimiento de datos de Escritura o de Salida, es decir, debe incluir un mínimo de dos movimientos de datos. No hay un límite superior para el número de movimientos de datos en un proceso funcional y, por lo tanto, no hay un límite superior para su tamaño (3.3)

k) Como una aproximación para fines de medición, los subprocesos de manipulación de datos no se miden por separado; se supone que la funcionalidad de cualquier manipulación de datos se debe al movimiento de datos con el que está asociada (3.5.6).

NOTA: El Modelo Genérico de Software de COSMIC, como su nombre lo indica, es un "modelo" lógico que expone las unidades en las que el software procesa datos que son adecuados para la medición de tamaño funcional. El modelo no pretende describir la secuencia física de los pasos por los que se ejecuta el software ni ninguna implementación técnica del software.

1.3.3. Tipos vs ocurrencias

Para comprender esta Manual de medición, es esencial que el lector pueda distinguir entre “tipos” y “ocurrencias”.

- En general, el "tipo" de una cosa es una clase abstracta de todas las cosas que comparten una característica común. (Los sinónimos de "tipo" son "categoría" o "clase"). En particular, en la Medición de tamaño funcional, las "cosas" son del mismo tipo si comparten el mismo FUR.
- Una "ocurrencia" de una cosa es cuando la cosa aparece en la práctica, por ejemplo, en un contexto del mundo real, o cuando ocurre un evento, o cuando un proceso es ejecutado por una persona o una computadora. (Un sinónimo de "ocurrencia" es "instancia"). Se crea una "ocurrencia" cuando a las características de un "tipo" de algo se les asignan valores reales, conocidos en el mundo orientado a objetos como "instanciamiento" (la creación de una instancia).

Todos los métodos de medición de tamaño funcional definen los "tipos de cosas" que un Medidor debe identificar en cualquier requisito funcional dado para medir un tamaño funcional.

En el caso del método COSMIC, estos "tipos de cosas" incluyen tipos de usuarios funcionales y tipos de procesos funcionales del Modelo Contextual de Software y todos los conceptos que se muestran en negrita, por ejemplo, tipos de movimiento de datos, tipos de objeto de interés, etc., en el Modelo Genérico de Software.

Sin embargo, para facilitar la lectura, normalmente omitimos "tipo" cuando usamos estos términos.

Ejemplos del Modelo Contextual de Software

EJEMPLO DE NEGOCIO 1: El sistema que soporta un Centro De Atención Telefónica tiene 100 empleados que responden las preguntas de los clientes. Un modelo contextual del sistema mostraría un tipo de usuario funcional: "Empleado del centro de llamadas" del cual hay 100 ocurrencias

EJEMPLO DE TIEMPO REAL 2: El software integrado de una radio digital envía su salida a un par de altavoces estéreo. El software envía señales separadas (del mismo tipo) a cada uno de los dos altavoces. Cada uno convierte la señal eléctrica recibida en sonido de la misma manera. Un modelo contextual del software mostraría un tipo de usuario funcional "altavoz" del cual hay dos apariciones.

Ejemplos del Modelo Genérico de Software

EJEMPLO DE NEGOCIO 3: Supongamos un proceso funcional (-tipo) que permite que los datos se ingresen y validen para un nuevo cliente. El movimiento de datos de Entrada (tipo) se ejecutará, es decir, ocurrirá una vez, cada vez que un usuario funcional humano registre datos para un nuevo cliente. Durante su ejecución, el proceso funcional debe validar los datos ingresados buscando para verificar si el cliente ya existe en la base de datos. Por lo tanto, el movimiento de datos de lectura (tipo) para esta validación ocurrirá una o más veces (según el diseño de la base de datos). Sin embargo, una entrada (tipo) y una lectura (tipo) del cliente se cuentan al medir este proceso.

EJEMPLO DE TIEMPO REAL 4: Supongamos un proceso funcional (tipo) que debe controlar la temperatura de un horno una vez cada diez segundos. El proceso funcional se ejecutará, es decir, ocurrirá, una vez cada 10 segundos. Durante su ejecución, el movimiento de salida (tipo) de datos del proceso para encender o apagar el calentador puede o no ocurrir, es decir, puede o no ocurrir en cualquier ciclo, dependiendo de si el calentador debe estar encendido o no. o apagado, o dejado en su estado actual. El movimiento de datos de salida (tipo) se cuenta una vez en el proceso funcional, independientemente de si se produce o no en una ejecución particular.

Nota: El número de ocurrencias de cualquier usuario funcional, o de cualquier movimiento de datos de Entrada (tipo), Salida (-tipo), Lectura (-tipo) o Escritura (-tipo) cuando se ejecuta el software es irrelevante para la medición de un tamaño funcional COSMIC. Sin embargo, determinar la frecuencia relativa de ocurrencias de movimiento de datos puede ayudar a distinguir diferentes tipos de movimiento de datos en ciertos casos. (Ver por ejemplo la regla en la sección 3.3.2).

1.4 El proceso de medición de software y la unidad de medida.

El proceso de medición COSMIC consta de tres fases:

- La fase de la estrategia de medición, en la que se definen el propósito y el alcance de la medición. El Modelo Contextual de Software se aplica entonces para que el software que se va a medir y la medición requerida se definan de manera inequívoca. (Capitulo 2)
- la Fase de Mapeo en la que el Modelo Genérico de Software se aplica al FUR del software que se medirá para producir el modelo COSMIC del software que se puede medir. (Capítulo 3)
- La fase de medición, en la que se miden los tamaños reales. (Capítulo 4)

Las reglas sobre cómo se deben registrar las mediciones se dan en el Capítulo 5.

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

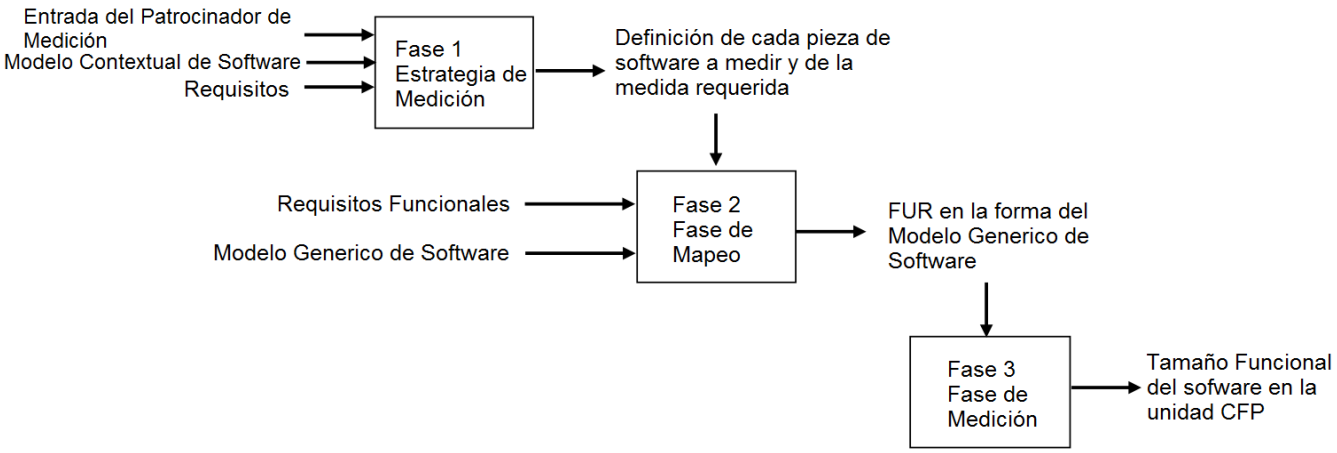


Figura 1.4 - El proceso de medición del método COSMIC

La unidad de medida COSMIC (la 'CFP') y el principio de medición se definen a continuación.

DEFINICION – Unidad de medida COSMIC
1 CFP (Cosmic Function Point) Puntos de Función COSMIC, que es el tamaño de un movimiento de datos.

PRINCIPIOS – El principio de medición COSMIC
a) El tamaño de un proceso funcional es igual al número de sus movimientos de datos.
b) El tamaño funcional de una pieza de software de alcance definido es igual a la suma de los tamaños de sus procesos funcionales.

Los tamaños de los cambios requeridos en una pieza de software se miden de la siguiente manera:

- El tamaño de cualquier movimiento de datos afectado (es decir, que debe agregarse, modificarse o eliminarse) por el cambio requerido se mide por convenio como 1 CFP.
- El tamaño de los cambios requeridos en una pieza de software es igual al número de movimientos de datos que se ven afectados por los cambios requeridos.
- El tamaño mínimo de un cambio en una pieza de software es 1 CFP.

En las secciones 4.1 a 4.4 de este Manual de medición se proporcionan normas y guías adicionales sobre la medición y la agregación de mediciones.

1.5 Limitaciones de la aplicabilidad del método COSMIC.

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

2

LA FASE DE LA ESTRATEGIA DE MEDICIÓN

2.0 Resumen del capítulo

Este capítulo describe los parámetros claves que deben considerarse en la primera fase “Estrategias de Medición” del proceso de medición, antes de comenzar realmente la medición. Estos parámetros esta (*en itálica*):

- El *propósito* de la medición, es decir, para qué se utilizará el resultado. El propósito determina los otros parámetros de una medida.
- El *alcance general* del software que se medirá y, si el software consta de más de una parte que debe medirse por separado (por ejemplo, los componentes de un sistema de software distribuido), el (los) alcance (s) de medición de las partes individuales.
- La *capa (s)* de la arquitectura del software en la que reside cada pieza del software que se va a medir.
- *El nivel de descomposición* de la (s) pieza (s) de software a medir, p. Ej. una aplicación completa o el nivel de sus componentes principales, o el nivel de componentes reutilizados de una Arquitectura Orientada a Servicios, etc.
- Los *usuarios funcionales* de cada pieza de software a medir. Estos son los remitentes y destinatarios de los datos hacia/desde el software que se va a medir; Pueden ser personas, dispositivos de hardware u otras piezas de software.
- El *nivel de granularidad* de los artefactos disponibles del software a medir. Por ejemplo, ¿todos los requisitos están definidos en el mismo nivel de detalle? ¿Tenemos suficientes detalles para una medición COSMIC precisa o solo lo suficiente para una medición aproximada?

La determinación de estos parámetros ayuda a responder las preguntas de "qué tamaño se debe medir", "con qué precisión queremos la medición", etc. El registro de los parámetros permite a los futuros usuarios de una medición decidir cómo interpretar la medición.

Es importante tener en cuenta que estos parámetros y conceptos relacionados no son específicos del método FSM de COSMIC, pero deben ser comunes a todos los métodos de Medición de Tamaño Funcional (FSM, por siglas en inglés -Functional Sizing Measurement-). Es posible que otros métodos de FSM no distingan diferentes tipos de usuarios funcionales y no discutan diferentes niveles de granularidad, etc. Es solo la aplicabilidad y flexibilidad más amplias del método COSMIC lo que requiere que estos parámetros se consideren más cuidadosamente que con otros métodos FSM.

Es muy importante registrar los datos que surgen de esta fase de la Estrategia de medición (como se indica en la sección 5.2) al registrar el resultado de cualquier medición. La falta de definición y registro de estos parámetros de manera consistente conducirá a mediciones que no se pueden interpretar y comparar de manera confiable, o se pueden usar de manera confiable como información para procesos como la estimación del esfuerzo del proyecto.

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

Cada sección proporciona una explicación básica de por qué el parámetro clave es importante, utilizando analogías para mostrar por qué el parámetro se da por sentado en otros campos de medición y, por lo tanto, también debe considerarse en el campo de la medición de tamaño funcional del software.

Observe en la Figura 2.0 que la determinación de los parámetros de la Estrategia de medición puede requerir cierta iteración.

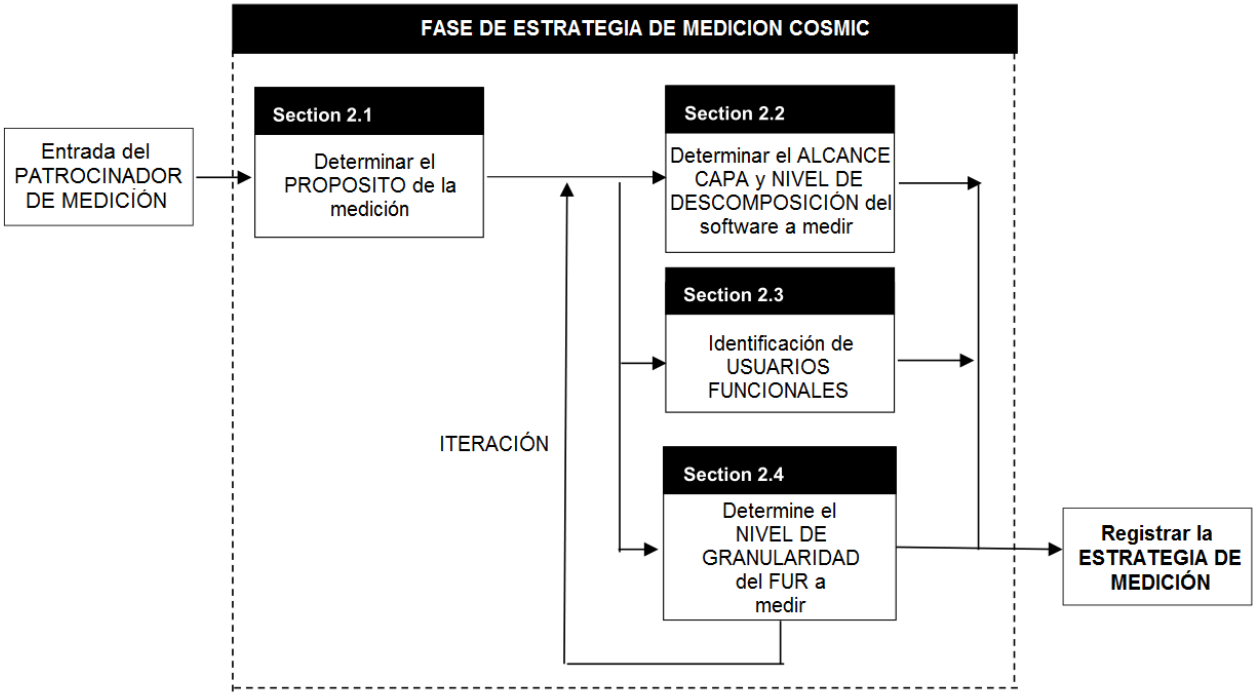


Figura 2.0 - El proceso de determinación de una estrategia de medición.

Patrones de Estrategia de Medición

Como ayuda para determinar una estrategia de medición, la Guía para "Patrones de estrategia de medición" [5] describe, para cada uno de los diferentes tipos de software, un conjunto estándar de parámetros para medir tamaños de software, denominado "patrón de estrategia de medición" (abreviado como "patrón de medición").

DEFINICIÓN – Patrón de medición (estrategia)
Una plantilla estándar que puede aplicarse al medir una parte del software de un dominio funcional del software dado, que define los tipos de usuarios funcionales 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 medición debería ayudar a los medidores a garantizar que las mediciones realizadas para el mismo propósito se realicen de manera coherente, se puedan comparar de forma segura con otras mediciones realizadas con el mismo patrón y se interpretarán correctamente para todos los usos futuros. Un beneficio adicional de usar un patrón estándar es que el esfuerzo para determinar los parámetros de la Estrategia de Medición se reduce mucho. Sin embargo, recomendamos enfáticamente que los Medidores estudien y dominen el método COSMIC, especialmente los parámetros de la Estrategia de Medición, antes de usar los patrones estándar.

2.1 Definición del propósito de la medida.

El término "propósito" se usa en su significado normal en inglés.

DEFINICIÓN – Propósito de medición
Una declaración que define por qué se requiere una medición y para qué se utilizará el resultado.

2.1.1 El propósito de medición – una analogía

Hay muchas razones para medir el tamaño funcional del software, al igual que hay muchas razones para medir, por ejemplo, las áreas de superficie de una casa y en ambos casos las diferentes razones pueden resultar en diferentes tamaños. A partir de la analogía, el tamaño de una casa puede variar con:

- la razón y el momento de la medición (por ejemplo, dependiendo de la necesidad de medir la especificación del esquema del cliente para el presupuesto, o los planes del arquitecto para una estimación de costo precisa, o el tamaño real al finalizar la planificación de los revestimientos de pisos),
- los artefactos medidos (por ejemplo, los planos o el edificio físico).

Tenga en cuenta, sin embargo, que se utilizan los mismos principios de medición y la unidad de medida para todas las mediciones. 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, dependiendo de la necesidad de medir antes del desarrollo con fines de estimación, o durante el desarrollo para realizar un seguimiento a la corrupción del alcance, o después de la instalación para medir el rendimiento del desarrollador),
- los artefactos medidos (por ejemplo, una declaración de requisitos, o los artefactos físicos del software).

Sin embargo, al igual que con la analogía, se utilizan los mismos principios de medición y la unidad de medida para todas las mediciones.

Claramente, el Medidor de una pieza de software debe decidir, dependiendo del propósito de la medición, *cuándo* medir (antes, durante o después del desarrollo), *qué* medir (por ejemplo, todo el software que debe entregar un proyecto o excluir software reutilizado) y *cuales artefactos* utilizar para obtener el FUR que se va a medir (por ejemplo, una declaración de requisitos o el software instalado).

EJEMPLOS: Los siguientes son propósitos típicos de medición

- *Medir el tamaño de los FUR a medida que evolucionan, como entrada a un proceso para estimar el esfuerzo de desarrollo.*
- *Medir el tamaño de los cambios en el FUR después de que se hayan acordado inicialmente, para administrar la "corrupción del alcance" del proyecto.*
- *Para medir el tamaño del FUR del software entregado como entrada para la medición del desempeño de la organización de desarrollo.*
- *Medir el tamaño del FUR del software total entregado, y también el tamaño del FUR del software que se desarrolló, para obtener una medida de reutilización funcional.*
- *Medir el tamaño del FUR del software existente como entrada para la medición del rendimiento del grupo responsable de mantener y respaldar el software.*
- *Para medir el tamaño de algunos cambios en (el FUR de) un sistema de software existente como una medida del tamaño de los resultados de trabajo de un equipo de proyecto de mejora.*
- *Para medir el tamaño del subconjunto de la funcionalidad total del software que debe desarrollarse, que se proporcionará a los usuarios funcionales humanos del software.*

2.1.2 La importancia del propósito

El propósito ayuda al medidor a determinar:

- El alcance que medir y, por tanto, los artefactos que serán necesarios para la medición.
- Los usuarios funcionales (como se mostrará en la sección 2.3, el tamaño funcional cambia según quien o que se define como el usuario funcional).
- El punto en el tiempo del ciclo de vida del proyecto cuando se llevará a cabo la medición.
- La precisión requerida de la medición y, por lo tanto, si se debe usar el método estándar de COSMIC, o se debe usar una versión aproximada del método estándar de COSMIC (por ejemplo, al principio del ciclo de vida de un proyecto, antes de que el FUR esté completamente elaborado)

Estos dos últimos puntos determinarán el nivel de granularidad en el que se medirá el FUR.

2.2 Definiendo el alcance de la medición.

DEFINICIÓN – Alcance de la medición
El conjunto de Requisitos Funcionales del Usuario que se incluirán en un ejercicio de medición de tamaño funcional especifico. NOTA: (específico para el método COSMIC) Se debe hacer una distinción entre el "alcance general", es decir, todo el software que debe medirse de acuerdo con el propósito, y el "alcance" de cualquier pieza de software individual dentro del alcance general, cuyo tamaño debe medirse por separado. En este Manual de medición, el término "alcance" (o la expresión "alcance de medición") se relacionará con 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 medir se derivará del propósito de la medición. b) El alcance de cualquier medición no debe extenderse a más de una capa del software a medir.

Consulte la siguiente sección para ver ejemplos del alcance general y los límites de medición.

2.2.1 Derivando el alcance de la medición del propósito de la medición.

El software definido por un *alcance general* puede subdividirse en piezas individuales de software, cada una con su propio *alcance de medición* definido de muchas maneras, dependiendo del propósito de la medición. Supongamos que un alcance general se define como "el portafolio de aplicaciones de la organización X" o como "todas las piezas de software que entregará el proyecto Y". Las subdivisiones se podrían hacer debido a:

- software en diferentes capas (debido a la regla b) anterior),
- diferentes responsabilidades organizativas, por ejemplo, por grupo de clientes o subproyectos,
- la necesidad de distinguir diferentes entregables para la medición del rendimiento, la estimación del esfuerzo o para los fines del contrato de software.

La última razón podría deberse a la necesidad de distinguir los ámbitos de medición separados para piezas de software que:

- se crean utilizando diferentes tecnologías, es decir, plataforma de hardware, lenguaje de programación, etc.
- operar en diferentes modos, es decir, en línea frente a modos por lotes,
- se desarrollan en lugar de "entregados" (este último incluye el paquete implementado u otro software reutilizado),
- están en diferentes niveles de descomposición, por ej. una aplicación completa o un componente principal o un componente secundario, como un objeto reutilizable,
- son los principales productos entregables en lugar del software que se usa una vez, por ejemplo, para la conversión de datos, y luego descartados; puede que no valga la pena el esfuerzo de medir este último,
- se entregan en un solo 'sprint' de un proceso ágil
- se desarrollan frente a software mejorado,

y cualquier combinación de estos factores.

En resumen, el propósito de la medición siempre debe utilizarse para determinar, (a) que software se incluye o excluye del alcance general y (b) la forma en que el software incluido debe dividirse en partes separadas, cada una con su propio alcance, para ser medido por separado.

En la práctica, una declaración de alcance debe ser explícita en lugar de genérica, por ejemplo, el resultado del trabajo desarrollado del equipo de proyecto "A", o la aplicación "B" o el portafolio de la empresa "C". La declaración de alcance también puede, por claridad, tener que declarar lo que está excluido.

EJEMPLO DE NEGOCIO: La Figura 2.1 muestra todas las piezas separadas del software -el "alcance global"- entregadas por un equipo de proyecto:

- *el cliente y los componentes del servidor de un paquete de software de aplicación implementado*
- *un programa que proporciona una interfaz entre el componente del 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 se creó utilizando una serie de objetos reutilizables desarrollados por el equipo del proyecto*
- *el software del controlador de dispositivo para el nuevo hardware en el que se ejecutará el componente cliente del paquete.*

(Cada pieza individual del software para la que se definió el alcance de medición se muestra como una caja rectangular sólida)

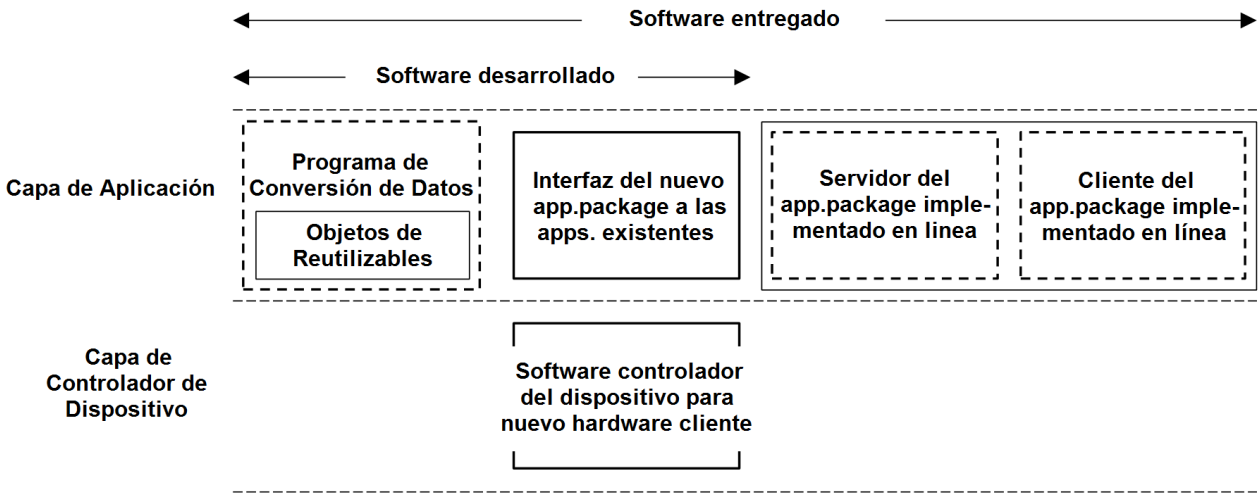


Figura 2.1 – El alcance general de los entregables de un proyecto de software y los límites de la medición individual.

El diagrama muestra que las piezas de software "entregadas" consistían en algunas que se desarrollaron recientemente y otras que fueron implementadas por el equipo del proyecto. El propósito es medir el FUR de las piezas individuales del software entregado que se agregarán al tamaño del portafolio de software de la organización, considerando el paquete de software en su totalidad, es decir, ignorando la estructura del componente cliente-servidor.

El tamaño del paquete implementado se agregó con el del programa de interfaz para actualizar el tamaño total del portafolio de aplicaciones de la organización. El tamaño del programa de conversión de datos no fue de interés ya que se usó una vez y se desechó. Pero el tamaño de cada uno de los objetos reutilizables se registró en el inventario de software de infraestructura de la organización, así como el del nuevo controlador del dispositivo. Nuevamente estos fueron clasificados por separado.

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

Tenga en cuenta también que normalmente solo es de interés medir el FUR del software resultante de la implementación de un paquete, no el FUR del paquete en sí. Este último es quizás solo de interés para el proveedor del paquete.

2.2.2 Capas

Dado a que el alcance de una pieza de software a medirse debe limitarse a una sola capa de software, el proceso de definición del alcance (s) de la medición puede requerir que el Medidor tenga primero que decidir cuáles son las capas de la arquitectura del software. En esta sección, por lo tanto, definiremos y discutiremos "capas" de software a medida que estos términos se usan en el método COSMIC. Las razones por las que necesitamos estas definiciones y reglas son las siguientes.

- El Medidor puede enfrentarse a la medición de algún software en un entorno "legado" de software que evolucionó durante muchos años sin haber sido diseñado de acuerdo con una arquitectura subyacente (el software que se medirá tiene la denominada "arquitectura de espagueti"). Por lo tanto, el Medidor puede necesitar orientación sobre cómo distinguir las capas de acuerdo con la terminología de COSMIC.
- Las expresiones "capa" y "arquitectura en capas" no se usan de manera consistente en la industria del software. Si el Medidor debe medir algún software que se describe como que está en una "arquitectura en capas", es recomendable verificar que las "capas" en esta arquitectura estén definidas de una manera que sea compatible con el método COSMIC. Para hacer esto, el Medidor debe establecer la equivalencia entre objetos arquitectónicos específicos en el paradigma de "arquitectura en capas" y el concepto de capas como se define en este manual.

Las capas pueden identificarse de acuerdo con las siguientes definiciones y principios.

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

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

PRINCIPIOS – Capa
a) El software en una capa proporciona un conjunto de servicios que es cohesivo según un criterio definido, y que el software en otras capas puede utilizar sin saber cómo se implementan esos servicios.
b) La relación entre el software en cualquiera de las dos capas se define por una "regla de correspondencia" que puede ser • "jerárquica", es decir, el software en la capa A puede usar 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 puede usar software en la capa B y viceversa.
c) El software en una capa intercambia los grupos de datos con el software en otra capa a través de sus respectivos procesos funcionales.
d) El software en una capa no usa necesariamente todos los servicios funcionales suministrados por el software en otra capa.
e) El software en una capa de una arquitectura de software definida puede dividirse en otras capas de acuerdo con una arquitectura de software definida diferente.

Una medición puede estar relacionada con dos o más piezas "pares" de software, definidas de la siguiente manera:

DEFINICIÓN – Piezas de software pares.
Dos piezas de software son iguales entre sí, si residen en la misma capa.

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

Si el alcance general del software que se va a medir se extiende sobre varias capas, el Medidor debe proceder de la siguiente manera.

- Si el software a medir existe dentro de una arquitectura de capas establecida que puede asignarse a los principios de capas de COSMIC como se definió anteriormente, entonces esa arquitectura debe usarse para identificar las capas para fines de medición.
- Sin embargo, si el propósito requiere que se mida algún software que no esté estructurado de acuerdo con los principios de capas de COSMIC, el Medidor debe tratar de dividir el software en capas aplicando los principios definidos anteriormente. Convencionalmente, los paquetes de software de infraestructura, como los sistemas de administración de bases de datos, los sistemas operativos o los controladores de dispositivos, que brindan servicios que pueden ser utilizados por otro software en otras capas, están ubicados en capas separadas.

Normalmente en las arquitecturas de software, la capa "superior", es decir, la capa que no está subordinada a ninguna otra capa en una jerarquía de capas se conoce como la capa "aplicación". El software en esta capa de aplicación se basa en los servicios del software en todas las otras capas para que funcione correctamente. El software en esta capa "superior" puede estar en capas, por ejemplo, como en una 'arquitectura de tres capas' de componentes de Interfaz de usuario, Reglas de negocios y Servicios de datos (consulte el Ejemplo de negocio 5 a continuación).

Una vez identificada, cada capa se puede registrar en la matriz del Modelo Genérico de Software (Apéndice A), con la etiqueta correspondiente.

EJEMPLO DE NEGOCIO 1: La estructura física de una arquitectura de software en capas típica (usando el término "capa" como se define aquí) que respalda el software de aplicaciones de negocios se muestra en la Figura 2.2:

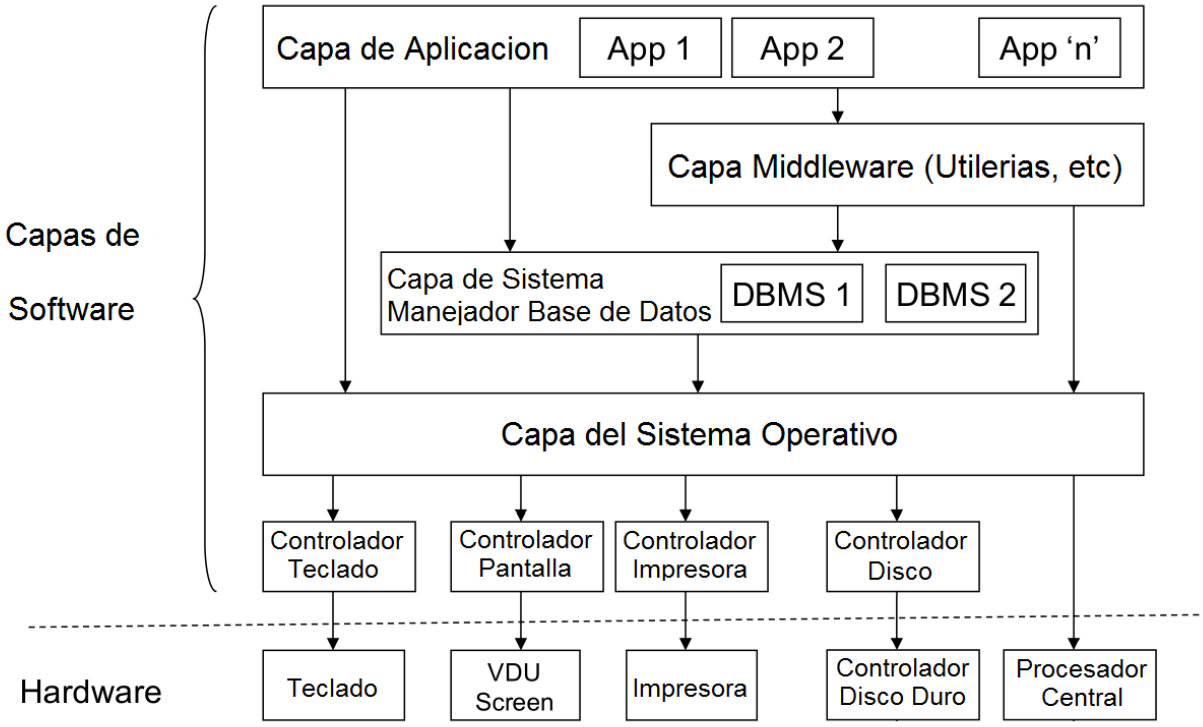


Figura 2.2- Arquitectura típica de software en capas para un sistema de negocios/MIS

EJEMPLO DE TIEMPO REAL 2: La estructura física de una arquitectura de software en capas típica (nuevamente utilizando el término 'capa' como se define aquí) que soporta una parte del software en tiempo real integrado se muestra en la Figura 2.3. Es posible que el software integrado en tiempo no necesite un sistema operativo en tiempo real.

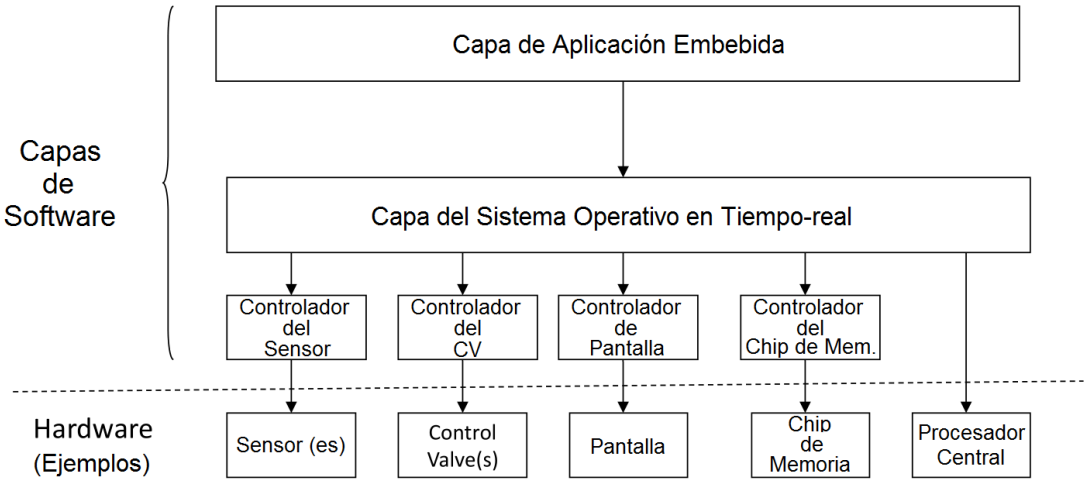


Figura 2.3 - Arquitectura típica en capas para un sistema de software integrado en tiempo real

EJEMPLO DE TIEMPO REAL 3: El modelo ISO de 7 capas (OSI) para telecomunicaciones. Esto define una arquitectura en capas para la cual las reglas de correspondencia jerárquicas para las capas del software de recepción de mensajes son el inverso de las reglas para las capas del software de transmisión de mensajes.

EJEMPLO DE TIEMPO REAL 4: la arquitectura "AUTOSAR" de la industria automotriz [19] que muestra todos los diferentes tipos de reglas de correspondencia entre capas que ahora se describen en los principios para una capa.

Una arquitectura de software puede exhibir diferentes capas dependiendo de la "vista" de la arquitectura.

EJEMPLO DE NEGOCIO 5: Considere una aplicación A situada en una arquitectura de software en capas, como se muestra en la Figura 2.4 a continuación, que muestra tres posibles estructuras de capas a), b) y c) de acuerdo con las diferentes vistas de arquitectura

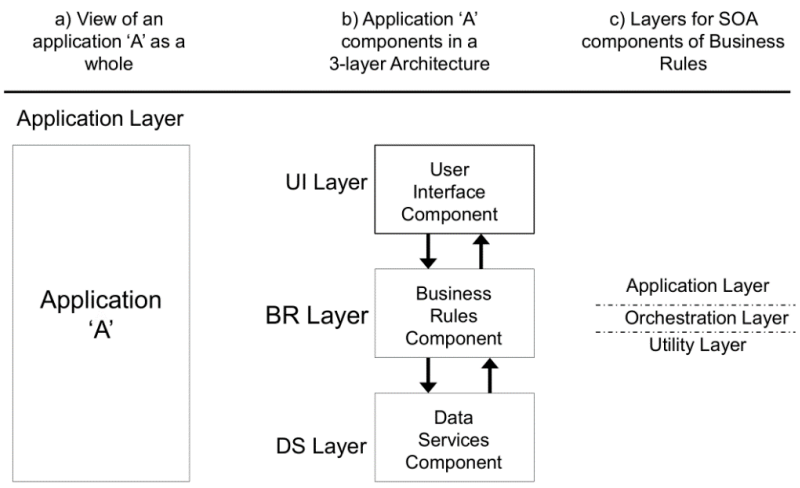


Figura 2.4 - Tres vistas de las capas de una aplicación

El propósito 1 es medir el tamaño funcional de la aplicación A 'como un todo', como en la Vista a). El alcance de medición es la totalidad de la aplicación A, que existe completamente dentro de la capa de "aplicación"

Propósito 2. La aplicación A se ha creado de acuerdo con una arquitectura de "tres capas" que comprende una interfaz de usuario, reglas de negocios y componentes de servicios de datos. El propósito 2 es medir los tres componentes por separado como en la Vista b). Cada componente está situado en su propia capa de la arquitectura y el alcance de la medición debe definirse por separado para cada componente.

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

2.2.3 Niveles de descomposición

El "nivel de descomposición" de un software se define de la siguiente manera:

DEFINICIÓN – Nivel de descomposición
Cualquier nivel resultante de dividir una pieza de software en componentes (llamado 'Nivel 1', por ejemplo), luego de dividir los componentes en subcomponentes ('Nivel 2'), luego de dividir los subcomponentes en sub-subcomponentes (' Nivel 3 '), etc. NOTA 1: No debe confundirse con el "nivel de granularidad" que se refiere al nivel de detalle de los requisitos. NOTA 2: Las medidas de tamaño de los componentes de una pieza de software solo son directamente comparables para los componentes en el mismo nivel de descomposición. NOTA 3: los diferentes niveles de descomposición de una parte del software pueden corresponder a diferentes "vistas" de las capas del software, por ejemplo, como en la Figura 2.4. Sin embargo, el software puede descomponerse en "niveles" independientemente de si está diseñado o no con un modelo de arquitectura en capas.

La Nota 2 en la definición anterior es importante porque los tamaños de piezas de software en diferentes niveles de descomposición no pueden sumarse simplemente sin tener en cuenta las reglas de agregación de la sección 4.3.1. Además, como consecuencia, el rendimiento (por ejemplo, la productividad = tamaño/esfuerzo) de los proyectos para desarrollar diferentes piezas de software solo se puede comparar de forma segura 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 completa', 'Componente mayor' y 'Componente menor'. Vea el ejemplo de negocios 5 en la sección 2.2.2 de este Manual de medición, donde los tres niveles se muestran en la Figura 2.4.

2.2.4 Definición del alcance de medición: resumen

Determinar el alcance de una medición puede implicar algo más que simplemente decidir qué funcionalidad debe incluirse en la medición. La decisión también puede implicar la consideración de la (s) capa (s) en la que reside el software que se medirá y el nivel de descomposición del software en el que se realizarán las mediciones, todo depende del propósito de la medición.

2.3 Identificando a los usuarios funcionales y reconociendo el almacenamiento persistente

Un "usuario" se define, en efecto[3], como "cualquier cosa que interactúe 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 determinada por los Requisitos funcionales de "usuario" que deben medirse

[3] Consulte el glosario para la definición, tomada de ISO / IEC 14143/1: 2007

y el propósito de la medición. Este (tipo de) usuario, conocido como el "usuario funcional", se define de la siguiente manera.

DEFINICIÓN – Usuario funcional.
Un (tipo de) usuario que se identifica en los Requisitos Funcionales del Usuario de una pieza de software que se mide como remitente y/o destinatario de los datos procesados por ese software.

La elección de los tipos de usuarios funcionales generalmente depende del dominio del software.

- En el dominio del software de aplicación de negocios, los usuarios funcionales son normalmente personas, además de otras aplicaciones similares con las que la aplicación interactúa.
- Para el software en tiempo real, los usuarios funcionales normalmente serían dispositivos de hardware diseñados para interactuar directamente con el software, además de otros programas similares con los que interactúa el software en tiempo real.

Nota: normalmente se recomienda encarecidamente NO medir el tamaño de la funcionalidad del software como lo ve una mezcla de personas y usuarios funcionales de dispositivos de hardware. El tamaño resultante será muy difícil de interpretar.

Tenga en cuenta que el conjunto total de "usuarios", es decir, incluyendo "cualquier cosa que interactúe con el software", debe incluir el sistema operativo. Pero el FUR de cualquier software de aplicación nunca incluiría el sistema operativo como usuario. Cualquier requisito que el sistema operativo pueda imponer en una aplicación será común a todas las aplicaciones, normalmente será manejado por el compilador o intérprete, será invisible para los usuarios funcionales reales de la aplicación y, por lo tanto, no aparecerá en el FUR. En la práctica medición de tamaño funcional, un sistema operativo nunca debe considerarse normalmente como un usuario funcional de una aplicación.

(Sin embargo, si el propósito es medir un componente de software de un sistema operativo como un controlador de dispositivo, entonces el sistema operativo que llama al controlador sería su usuario funcional).

REGLAS – Usuarios funcionales
a) Los usuarios funcionales de una pieza de software a medir dependerán del propósito de la medición.
b) Cuando el propósito de una medición de una pieza de software se relaciona con el esfuerzo por desarrollar o modificar la pieza de software, los usuarios funcionales deben ser todos los diferentes tipos de remitentes y/o destinatarios de datos deseados a/desde la funcionalidad nueva o modificada, según lo requerido por su FUR.
NOTA: FUR puede especificar que múltiples apariciones de usuarios funcionales deben identificarse individualmente. Sin embargo, serán del mismo tipo si cada aparición está sujeta a la misma FUR.

EJEMPLO DE NEGOCIO 1: que ilustra la regla b): En un sistema de pedidos, varios empleados (usuarios humanos funcionales) mantienen los datos del pedido. El ID de empleados se agrega a todos los grupos de datos que ingresan. Identifique un tipo de usuario funcional "empleado" porque los FUR del sistema de pedido son los mismo para todos estos empleados.

EJEMPLO DE TIEMPO REAL 2: que ilustra la regla b): Cada rueda de un automóvil tiene un sensor que obtiene la presión de su neumático. A intervalos regulares, un proceso funcional debe obtener la presión de los cuatro neumáticos. Si la presión es demasiado baja o alta (el rango de presiones seguras está en el software), el software muestra qué neumático tiene un problema de presión en un diagrama de las cuatro ruedas en una pantalla de visualización en el tablero de instrumentos. Los usuarios funcionales son los cuatro sensores y la pantalla de visualización. Sin embargo, los cuatro sensores son del mismo tipo, así que identifique un tipo de usuario funcional "sensor" y un tipo de usuario funcional para la pantalla de visualización.

Nota: los sensores de presión de los cuatro neumáticos funcionan de la misma manera; El tipo de grupo de datos que envían es el mismo y el procesamiento de estos datos es idéntico. Obviamente, el software debe poder distinguir los cuatro neumáticos, (por ejemplo, utilizando la secuencia en la que informan las presiones de los neumáticos cuando se solicitan, o por su ID (por ejemplo, 1 a 4) que envían con su presión, etc.) para para mostrar en la pantalla la presión asociada a cada neumático. Pero el FUR para el procesamiento de datos de cada neumático es el mismo, por lo que solo hay un tipo de usuario funcional: "neumático".

2.3.1 El tamaño funcional puede variar con los usuarios funcionales.

No siempre es el caso que los usuarios funcionales sean obvios. Los diferentes tipos de usuarios de una "cosa" pueden "ver" una funcionalidad diferente y, por lo tanto, pueden medir diferentes tamaños de la "cosa". En el caso del software, los diferentes (tipos de) usuarios funcionales pueden requerir (a través de su FUR) una funcionalidad diferente y, por lo tanto, los tamaños funcionales variarán con la elección de los usuarios funcionales.

EJEMPLO DE NEGOCIO 1: Un sistema de software tiene una funcionalidad para mantener datos personales básicos accesibles para todo el personal del Departamento de Personal, y datos de salarios más sensibles accesibles solo para un subconjunto de este personal. Así que este software tiene dos tipos de usuarios funcionales. El propósito de una medición de este software debe definir si el alcance de la medición se aplica a su funcionalidad total accesible para todo el personal o si está restringido a la funcionalidad disponible para uno de los dos tipos de personal.

EJEMPLO DE TIEMPO REAL 2: Considere el software integrado de una copiadora. Excluyendo el sistema operativo de la copiadora como posible usuario funcional, los usuarios funcionales del software podrían definirse de una de dos maneras. Podrían ser (a) el usuario humano que desea hacer copias, o (b) los dispositivos de hardware de la copiadora, es decir, los botones de control, una pantalla en la que se muestran los mensajes al usuario humano, el mecanismo de transporte de papel, los sensores de atasco de papel, el controlador de tinta, luces indicadoras, etc., con las que el software interactúa directamente. Estos dos tipos de usuarios funcionales, los humanos o el conjunto de dispositivos de hardware, "verán" una funcionalidad diferente. El usuario humano, por ejemplo, conocerá solo un subconjunto de la funcionalidad total del software de copiadora.

Los desarrolladores del software integrado que impulsa la copiadora deberán definir los dispositivos de hardware como sus usuarios funcionales. Alternativamente, a una persona de mercadeo le puede resultar útil medir el tamaño de la funcionalidad de la copiadora de su propia compañía según lo ve un usuario funcional humano en comparación con el producto de un competidor para comparar su precio/rendimiento[4]. NO intente mezclar los dos. puntos de vista; una medición de tamaño desde una vista "mixta" de humano/hardware sería muy difícil de interpretar.

Una vez identificados los usuarios funcionales, es sencillo identificar el límite. El límite se encuentra entre la pieza de software que se mide y sus usuarios funcionales. Ignoramos cualquier otro hardware o software en ese espacio intermedio[5].

DEFINICIÓN – Frontera.
Una interfaz conceptual entre el software que se mide y sus usuarios funcionales. NOTA: De la definición se desprende que hay una frontera entre dos piezas de software en la misma capa o en capas diferentes que intercambian datos donde una pieza de software es un usuario funcional de la otra y/o viceversa.

NOTA: Esta definición de "límite" se toma de ISO/IEC 14143/1: 2007, modificada por la adición de "funcional" para calificar "usuario". Para evitar la ambigüedad, tenga en cuenta que la frontera no debe confundirse con ninguna línea que pueda dibujarse alrededor de algún software que se medirá para definir el alcance de la medición. El límite no se utiliza para definir el alcance de una medición.

[4] Toivonen, por ejemplo, comparó el tamaño de la funcionalidad de los teléfonos móviles disponibles solo para usuarios humanos en "Definición de medidas para la eficiencia de la memoria del software en terminales móviles", Taller internacional sobre medición de software, Magdeburgo, Alemania, octubre de 2002.

[5] De hecho, si el medidor ha tenido que examinar el FUR para identificar a los remitentes y destinatarios de los datos, el límite ya habrá sido identificado.

2.3.2 Almacenamiento Persistente.

DEFINICIÓN – Almacenamiento Persistente.
Almacenamiento que permite que un proceso funcional almacene un grupo de datos más allá de la vida del proceso funcional y/o desde el 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 algún otro proceso. NOTA 1: En el modelo COSMIC, el almacenamiento persistente es un concepto que existe solo dentro del límite del software que se mide, por lo que no puede considerarse como un usuario funcional del software que se está midiendo. NOTA 2: Un ejemplo de "algún otro proceso" sería la fabricación de memoria de solo lectura.

(Para la definición de un "proceso funcional", consulte la sección 3.2)

La mayoría de las veces, cuando se analiza el FUR en busca de algún software, si el Medidor encuentra los requisitos para almacenar datos en, o para recuperar datos de un archivo o base de datos específico, el FUR no está interesado en dónde o cómo se almacena físicamente el archivo/base de datos; Los últimos son requisitos técnicos. Todo lo que FUR debe aclarar es que los datos están almacenados. En el Modelo Genérico de Software, el lugar donde se almacenan los datos es el "almacenamiento persistente". El almacenamiento persistente no necesita ser identificado; Está disponible para el software en cualquier capa.

Una excepción a esta guía general es si el FUR del software que se está midiendo especifica que los datos se muevan directamente hacia o desde un dispositivo de almacenamiento de hardware físico identificable. ("Directamente" significa "sin pasar por ningún software intermedio".) Si el software que se mide directamente interactúa con un dispositivo de hardware, el dispositivo debe ser un usuario funcional de ese software; No debe considerarse como almacenamiento persistente. Para más sobre esto, vea la sección 3.5.8.

2.3.3 Diagramas de contexto.

Puede ser muy útil al definir un alcance de medición y los usuarios funcionales dibujar un "diagrama de contexto" para el software que se está midiendo. En esta y en otras guías de COSMIC, los diagramas de contexto se utilizan para mostrar el alcance de una pieza de software que debe medirse dentro de su contexto de usuarios funcionales (personas, dispositivos de hardware u otro software) y los movimientos de datos entre ellos. (Los diagramas de contexto también suelen mostrar almacenamiento persistente, si es relevante).

Un diagrama de contexto es efectivamente una instancia de un patrón de medición (consulte la sección 2.0) aplicado al software que se está midiendo. La clave de los símbolos utilizados en los diagramas de contexto es como se indica en la Figura 2.5:

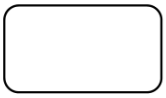
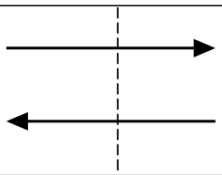
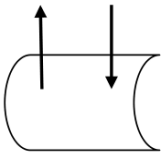
Símbolo	Interpretación
	La pieza de software a medir (caja con contorno grueso), es decir, la definición de un alcance de medición.
	Cualquier usuario funcional del software que se esté midiendo.
	Las flechas representan **todo** el movimiento de los datos que cruzan un límite (la línea de puntos) entre un usuario funcional y el software que se está midiendo
	Las flechas representan todos los movimientos de datos entre el software que se mide y el "almacenamiento persistente". (El símbolo del diagrama de flujo estándar para "almacenamiento de datos" enfatiza 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 - Clave de los símbolos de los diagramas de contexto.

EJEMPLO DE NEGOCIO: La Figura 2.6 muestra el diagrama de contexto para el software cliente/servidor del paquete de aplicación implementado como en el Ejemplo que se muestra en la Figura 2.1 de la sección 2.2.1, que debe medirse como un 'todo', es decir, el hecho de que el paquete de la aplicación tiene dos componentes (cliente y servidor) que deben ignorarse para esta medición del paquete.

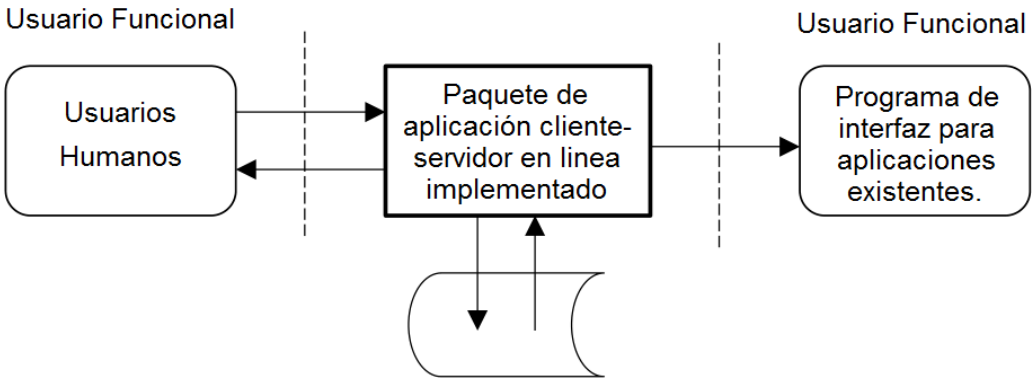


Figura 2.6 - Diagrama de contexto para la aplicación cliente-servidor de la sección 2.2.1

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

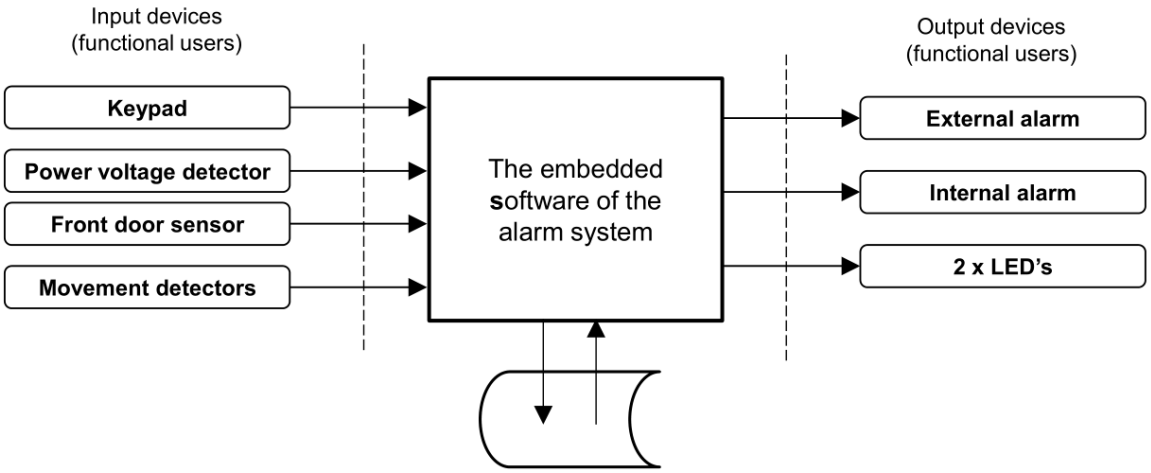


Figura 2.7 - Diagrama de contexto para el software incorporado de un sistema de alarma contra intrusos

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 actuales se especifican "a un alto nivel", es decir, en resumen, o con poco detalle. A medida que el proyecto avanza, los requisitos actuales se refinan (por ejemplo, a través de las versiones 1, 2, 3, etc.), revelando cada vez más detalles "a niveles más bajos". Estos diferentes grados de detalle de los requisitos actuales se conocen como "niveles de granularidad" diferentes. (Consulte también la sección 2.4.2 para conocer otros términos que pueden confundirse con el concepto de "nivel de granularidad" tal como se define aquí).

DEFINICIÓN – Nivel de granularidad
Cualquier nivel de expansión de la descripción de cualquier parte 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 incrementado de expansión, la descripción de la funcionalidad la pieza de software se encuentra en un nivel de detalle aumentado y uniforme. NOTA: los medidores deben tener en cuenta que cuando los requisitos evolucionan al inicio de la vida de un proyecto de software, en cualquier momento, diferentes partes de la funcionalidad de software requerida normalmente se documentarán a diferentes niveles de granularidad.

En la mayoría de las actividades de desarrollo de productos, los planes se dibujan a escalas estándar, y es fácil traducir las dimensiones medidas en un dibujo a las de otro dibujo con una escala diferente. En contraste, no hay escalas estándar para los diversos niveles de granularidad en los que se puede especificar el software, por lo que puede ser difícil estar seguro de que dos declaraciones de requisitos funcionales se encuentran en el mismo nivel de granularidad. Sin un acuerdo sobre algún nivel estándar de granularidad en el cual medir (o a qué medidas se deben escalar), es imposible saber con certeza que se pueden comparar dos medidas de tamaño funcional.

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

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

Si no reconociéramos el fenómeno de los diferentes niveles de granularidad, parecería que estos tres mapas revelaron diferentes tamaños de la red de carreteras de la nación. Por supuesto, con los mapas de ruta, todos comprenden los diferentes niveles de detalle que se muestran y hay escalas estándar para interpretar el tamaño de la red revelada en cualquier nivel. El concepto abstracto de "nivel de granularidad" se encuentra detrás de las escalas de estos diferentes mapas.

Para la medición de software, solo hay un nivel estándar de granularidad que es posible definir de forma inequívoca. Ese es el nivel de granularidad en el que los procesos funcionales individuales y sus movimientos de datos se pueden identificar y definir. Las mediciones deben hacerse en este nivel o escalarse a este nivel siempre que sea posible[6].

2.4.2 Aclaración del "nivel de granularidad"

Antes de seguir adelante, es importante asegurarse de que no haya malentendidos sobre el significado de "nivel de granularidad" en el método COSMIC. Acercar los requisitos funcionales implica ampliar la *descripción* de algunos programas desde un nivel de granularidad "superior" a uno "inferior" y revelar más detalles, *sin cambiar su alcance*. Este proceso NO debe confundirse con ninguno de los siguientes.

- Hacer zoom en algún software para revelar sus componentes, subcomponentes, etc. (a diferentes "niveles de descomposición", consulte la sección 2.2.3 anterior). Tal acercamiento puede ser requerido si el propósito de la medición requiere que el alcance general de la medición esté subdividido siguiendo la estructura física del software.
- Evolucionar la descripción de algún software a medida que avanza a través de su ciclo de desarrollo, por ejemplo. desde los requisitos hasta el diseño lógico, el diseño físico, etc. Sea cual sea la etapa en el desarrollo de algún software, solo nos interesa su FUR para fines de medición.

Por lo tanto, el concepto de "nivel de granularidad" debe interpretarse como aplicable solo a la descripción de los requisitos del software.

2.4.3 El nivel de granularidad estándar del proceso funcional.

El "nivel de granularidad del proceso funcional" es el único nivel de granularidad de los requisitos funcionales que es posible definir como estándar. Desde este nivel, podemos definir un conjunto de conceptos que pueden resultar en mediciones de tamaño repetibles y, por lo tanto, comparables.[7]

DEFINICIÓN – Nivel de granularidad de los procesos funcionales
Cualquier nivel de granularidad en la descripción de un software en el que: • sus usuarios funcionales (-tipos) son humanos individuales o dispositivos de ingeniería o piezas de software (y no un grupo de estos) Y • evento único (-tipos) ocurre que el software debe responder (y no a cualquier nivel de granularidad en el que se definen los grupos de eventos). NOTA 1: En la práctica, la documentación del software a menudo describe los requisitos funcionales de diferentes partes del software a diferentes niveles de granularidad, especialmente cuando la documentación aún está en evolución. Los procesos funcionales pueden revelarse en cualquiera de estos diferentes niveles de granularidad. NOTA 2: 'Grupos de estos' (usuarios funcionales) podrían, por ejemplo, ser 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: 'Grupos de eventos' podría, por ejemplo, indicarse en una declaración de requisitos funcionales con un alto nivel de granularidad mediante una secuencia de entrada a un sistema de software de contabilidad etiquetado como 'transacciones de ventas'; o por un flujo de entrada a un sistema de software de aviónica etiquetado como 'comandos piloto'.

[6] El tema de la escala de mediciones de un nivel de granularidad a otro se trata en la "Guía para la Medición del Tamaño Funcional COSMIC Temprana o Rápida por enfoque de aproximación" [6].

[7] La razón para el nombre "nivel de granularidad de los procesos funcional" es que este es el nivel en el que se identifican los procesos funcionales; consulte la sección 3.2 para una discusión más detallada de los procesos funcionales.

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

REGLAS – Nivel de granularidad para medir un proceso funcional
a) Una medición de tamaño funcional de una pieza de software requiere que sus FUR sean conocidos en los niveles de granularidad en los que se pueden identificar sus procesos funcionales y sus subprocesos 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 pueden medirse utilizando un enfoque aproximado. Estos enfoques definen cómo los requisitos pueden medirse a niveles más altos de granularidad. Luego, estos factores de escala se aplican a las mediciones en los niveles más altos de granularidad para producir un tamaño aproximado en los niveles de granularidad de los procesos funcionales y sus subprocesos de movimiento de datos. Consulte la "Guía para la Medición del Tamaño Funcional COSMIC Temprana o Rápida por enfoque de aproximación". [6]

Además de las reglas, COSMIC recomienda[8] que el nivel de granularidad en el que se conocen los procesos funcionales y sus subprocesos de movimiento de datos debe ser el estándar en el que los proveedores de servicios de evaluación comparativa y herramientas de software diseñados para usar y medir herramientas de software diseñen apoyar o usar medidas de tamaño funcional, p. ej. para estimar el esfuerzo del proyecto.

EJEMPLO DE NEGOCIO: El ejemplo, del dominio del software de aplicación de negocios, es parte de un sistema bien conocido para ordenar productos a través de Internet, que llamaremos "Aplicación de pedido de Everest". El propósito de este ejemplo es ilustrar diferentes niveles de granularidad y que los procesos funcionales pueden revelarse en diferentes niveles. La descripción a continuación es altamente simplificada para los fines de esta ilustración de los niveles de granularidad.

Si quisiéramos medir esta aplicación, podríamos asumir que el propósito de la medición es determinar el tamaño funcional de la parte de la aplicación disponible para los usuarios humanos clientes (como "usuarios funcionales"). Luego definiríamos el alcance de la medición como "las partes de la aplicación de Everest a las que pueden acceder los clientes para realizar pedidos a través de Internet". Sin embargo, tenga en cuenta que el propósito de este ejemplo es ilustrar diferentes niveles de granularidad. Por lo tanto, exploraremos solo algunas partes de la funcionalidad total del sistema que es suficiente para comprender este concepto de niveles de granularidad. Este ejemplo trata sobre los niveles de granularidad de la FUR; no dice nada sobre cualquier posible descomposición del software subyacente.

En el más alto "Nivel 1 (Función principal)" de esta parte de la aplicación, una declaración de los requisitos de la Aplicación de pedidos Everest sería una declaración resumida simple como la siguiente.

"La aplicación de pedidos de Everest debe permitir a los clientes consultar, seleccionar, ordenar, pagar y obtener la entrega de cualquier artículo de la gama de productos de Everest, incluidos los productos disponibles de proveedores externos".

Acercándonos a esta declaración de alto nivel de los requisitos, encontramos que en el siguiente nivel inferior 2, la aplicación de pedidos Everest consta de cuatro funciones secundarias, como se muestra en la Figura 2.8 (a).

[8] La razón por la que el uso del nivel de granularidad del proceso funcional se 'recomienda' en lugar de proporcionarse como una regla es que esta recomendación se aplica no solo a los usuarios individuales del método COSMIC sino a sus redes de proveedores de servicios y herramientas que usan las medidas de tamaño. COSMIC solo puede hacer recomendaciones a esta comunidad más amplia.

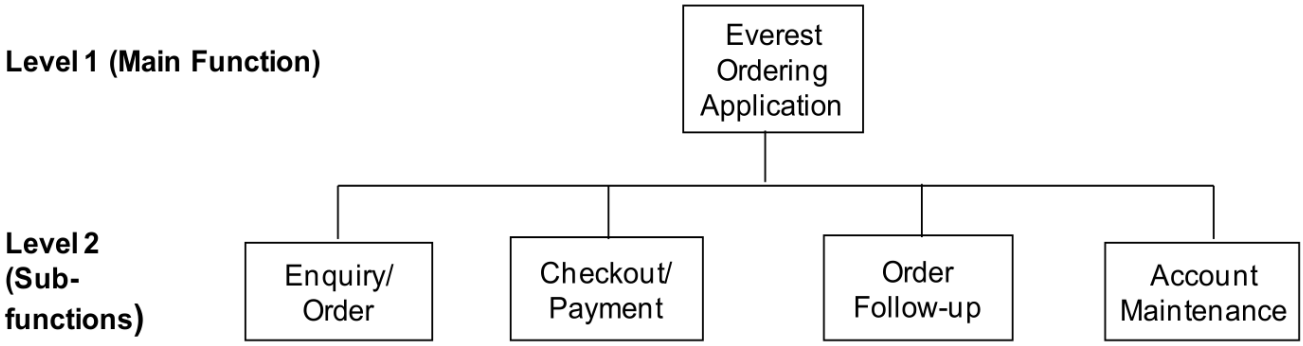


Figura 2.8 (a) - Análisis del sistema de pedido de Everest: los dos primeros niveles de granularidad

Los requisitos de las cuatro subfunciones son:

- *La subfunción de Consulta/Pedido, que permite a un cliente encontrar cualquier producto en la base de datos de Everest, así como su precio y disponibilidad, y agregar cualquier producto seleccionado a una "cesta" para la compra. Esta subfunción también promueve las ventas al sugerir ofertas especiales, ofrecer revisiones de artículos seleccionados y permitir consultas generales, como las condiciones de entrega, etc. Es una subfunción muy compleja. Por lo tanto, no analizamos esta subfunción con más detalles por debajo del nivel 2 para los propósitos de este ejemplo.*
- *La subfunción Checkout/Payment (Pago/Pago) que permite a un cliente comprometerse a ordenar y pagar los bienes en la canasta.*
- *La subfunción de seguimiento de pedidos, que permite a un cliente preguntar cuánto ha progresado un pedido existente en el proceso de entrega, mantener su pedido (por ejemplo, cambiar la dirección de entrega) y devolver bienes no satisfactorios.*
- *La subfunción de mantenimiento de la cuenta que permite a un cliente existente mantener varios detalles de su cuenta, como la dirección de la casa, los medios de pago, etc.*
- *Las figuras 2.8 (b) y (c) muestran algunos detalles que se revelan cuando ampliamos los requisitos, un nivel adicional de granularidad en la subfunción de pago/pago, la subfunción de seguimiento de la orden y la cuenta Subfunción de mantenimiento. En este proceso de zoom es importante tener en cuenta que*
- *no hemos cambiado el alcance de la funcionalidad a medir, y*
- *todos los niveles de la descripción de la aplicación Everest muestran la funcionalidad disponible para los clientes (como usuarios funcionales). Un cliente puede "ver" la funcionalidad de la aplicación en todos estos niveles de granularidad.*

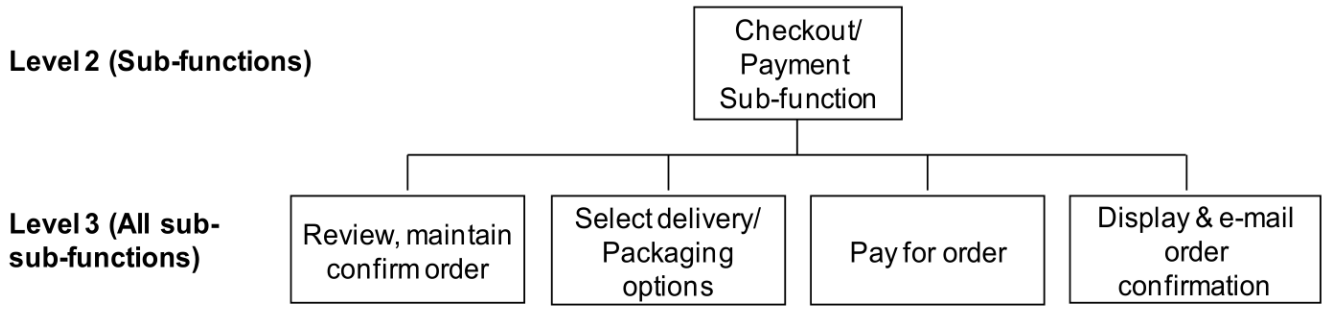


Figura 2.8 (b). Análisis de la subfunción Pago/Checkout

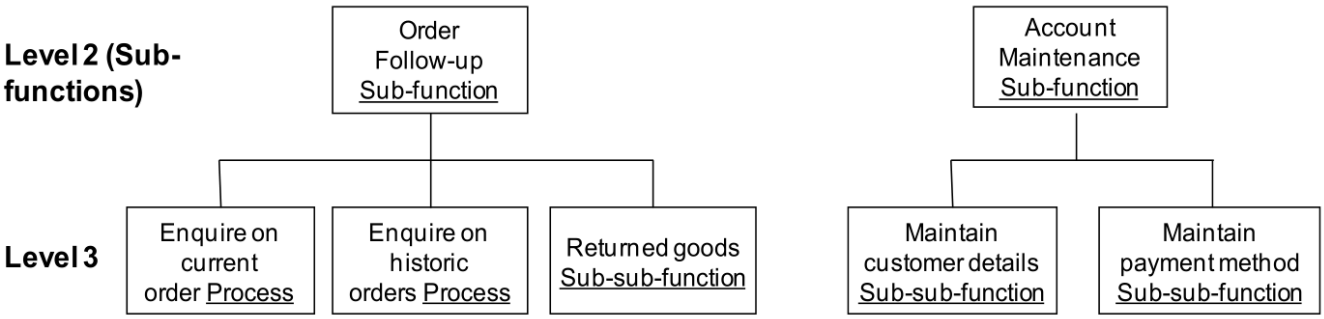


Figura 2.8 (c). Análisis e la subfunción de Seguimiento de Orden y Mantenimiento de cuentas

La Figura 2.8 (c) ahora revela que cuando nos acercamos al nivel inferior 3 de este análisis particular de la subfunción de Seguimiento de la Orden, encontramos dos procesos[9] funcionales individuales en el nivel 3 (para dos consultas en la subfunción Seguimiento de Orden). Se revelarían más procesos funcionales si continuáramos el refinamiento de las sub-subfunciones del Nivel 3 a niveles más bajos. Este ejemplo demuestra, por lo tanto, que cuando alguna funcionalidad se refina en un enfoque 'de arriba hacia abajo', no se puede suponer que la funcionalidad mostrada en un 'nivel' particular en un diagrama siempre corresponderá al mismo 'nivel de granularidad' que este concepto se define en el método COSMIC. (Esta definición requiere que, en cualquier nivel de granularidad, la funcionalidad esté "en un nivel de detalle comparable").

Además, otros analistas podrían dibujar los diagramas de manera diferente, mostrando otros grupos de funcionalidades en cada nivel del diagrama. No hay una forma "correcta" de acercarse a la funcionalidad de un sistema tan complejo.[10]

Dadas estas variaciones que inevitablemente ocurren en la práctica, un Medidor debe examinar cuidadosamente los distintos niveles de un diagrama de análisis para encontrar los procesos funcionales que deben medirse. Donde en la práctica esto no es posible, por ejemplo, debido a que el análisis aún no ha alcanzado el nivel donde se han revelado todos los procesos funcionales, se debe aplicar la regla (b) anterior. Para ilustrar esto, examinemos el caso de la "Subfunción de detalles del cliente" (consulte la Figura 2.8 (c) más arriba), en la rama de la subfunción de Mantenimiento de la cuenta.

Para un Medidor experimentado, la palabra "mantener" casi invariablemente sugiere un grupo de eventos y por lo tanto un grupo de procesos funcionales. Por lo tanto, podemos asumir que esta subfunción "Mantener" debe comprender tres procesos funcionales, a saber, una "consulta sobre los detalles del cliente", "actualizar los detalles del cliente" y "eliminar los detalles del cliente". (El proceso de "crear detalles del cliente" también debe existir, obviamente, pero esto ocurre en otra rama del sistema, cuando un cliente pide productos por primera vez. Está fuera del alcance de este ejemplo simplificado).

Un Medidor con experiencia debe poder "estimar" un tamaño de esta subfunción en unidades de Puntos de Función COSMIC tomando el número supuesto de procesos funcionales (tres en este caso) y multiplicando este número por el tamaño promedio de un funcional proceso. Este tamaño promedio se obtendría por calibración en otras partes de este sistema o en otros sistemas comparables. Los ejemplos de este proceso de calibración se proporcionan en el documento "Guía para la Medición del Tamaño Funcional COSMIC Temprana o Rápida por enfoque de aproximación" [6] que también contiene otros ejemplos de otras aproximaciones para el dimensionamiento aproximado.

Claramente, tales métodos de aproximación tienen sus limitaciones. Si aplicamos dicho enfoque a la declaración de requisitos del Nivel 1 como se indicó anteriormente ('La aplicación de Everest debe permitir a los clientes consultar, seleccionar, ordenar, pagar y obtener la entrega de cualquier artículo de la gama de productos de Everest ...'), podríamos identificar algunos procesos funcionales. Pero un análisis más detallado revelaría que el número real de procesos funcionales en esta aplicación compleja debe ser mucho mayor. Por lo que los tamaños funcionales por lo general parecen aumentar a medida que se establecen más detalles de los requisitos, incluso sin cambios en el alcance. Por lo tanto, estos métodos de

[9] (Nota: en esta etapa, alguien que sea nuevo en el método COSMIC puede no estar convencido sobre la base de la definición y las reglas para el 'nivel de granularidad del proceso funcional' de que las dos consultas que han aparecido en el nivel 3 son en realidad procesos funcionales, a diferencia de los dos subsistemas que podrían descomponerse aún más. La Sección 3.2 sobre 'Identificación de procesos funcionales' proporcionará más evidencia para respaldar el análisis que se presenta aquí.

[10] La Figura 2.8 puede no ser un ejemplo de la mejor práctica, pero es típico de cómo se pueden dibujar tales diagramas.

aproximación deben utilizarse con gran cuidado en niveles altos de granularidad, cuando hay muy pocos detalles disponibles.

Para ver un ejemplo de medición a diferentes niveles de granularidad y de descomposición, consulte el ejemplo del sistema de telecomunicaciones en la "Guía para la Medición del Tamaño Funcional COSMIC Temprana o Rápida por enfoque de aproximación"

2.5 Observaciones finales sobre la fase de la estrategia de medición

Es esencial determinar y documentar los parámetros de la estrategia de medición para garantizar que el tamaño resultante se pueda entender y utilizar correctamente en el futuro. La gran mayoría de las mediciones de tamaño funcional se llevan a cabo con un propósito que está relacionado de alguna manera con el esfuerzo de desarrollo, por ejemplo. para la medición del desempeño del desarrollo del proyecto, o para la estimación del proyecto. En muchas de estas situaciones, se puede usar un patrón de medición estándar: consulte la “Guía sobre patrones de estrategia de medición” [5]

3

LA FASE DE MAPEO

3.0 Resumen del capitulo

Este capítulo analiza la segunda fase, "Mapeo", del proceso de medición definiendo los conceptos clave del Modelo Genérico de Software y el proceso a seguir para mapear el FUR del software al modelo, de modo que el FUR pueda medirse. Estos conceptos clave del Modelo Genérico de Software son: *(en cursiva):*

- Un evento hace que un *usuario funcional* solicite un servicio del software que se está midiendo. Dicho evento se denomina *"evento desencadenante"* y el servicio solicitado se denomina *"proceso funcional".*
- Los procesos funcionales se componen de dos tipos de subprocesos que mueven datos *("movimientos de datos")* o que manipulan datos *("manipulación de datos")*. Los subprocesos de manipulación de datos no se reconocen por separado, pero se consideran responsables de los movimientos de datos con los que están asociados.
- Un movimiento de datos mueve un *"grupo de datos"*. Un grupo de datos consta de *atributos* de datos que describen un *"objeto de interés",* es decir, una *"cosa"* que es de interés en el mundo del usuario funcional en cuestión.
- Existen cuatro tipos de movimientos de datos: las *Entradas* y las *Salidas* mueven un grupo de datos dentro y fuera de un proceso funcional a través de un *límite o frontera* desde / hacia un usuario funcional, respectivamente. Las *Lecturas* y *Escrituras* mueven 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 y reglas integrales para ayudar a identificar los conceptos correctamente, y ejemplos extensos de los conceptos para diferentes tipos de software.

3.1 Mapeo de FUR al Modelo Genérico de Software

La Figura 3.0 muestra los pasos del proceso para mapear los Requisitos Funcionales del Usuario (FUR) como en los artefactos de software disponibles al formulario requerido por el Modelo Genérico de Software COSMIC.

Cada paso en este proceso es el tema de una sección específica indicada en la barra de título del paso en la Figura 3.0.

El proceso está diseñado para ser aplicable a una amplia gama de artefactos de software. Alentamos a los Medidores a usar este proceso general para derivar reglas más específicas para su uso en su entorno local para mapear desde los artefactos de software producidos por su método de ingeniería de software local hasta el Modelo Genérico de Software COSMIC. El objetivo de un proceso local específico, ilustrado con ejemplos locales, debería ser reducir la incertidumbre en el mapeo y, por lo tanto, mejorar la precisión y la repetibilidad de las mediciones.

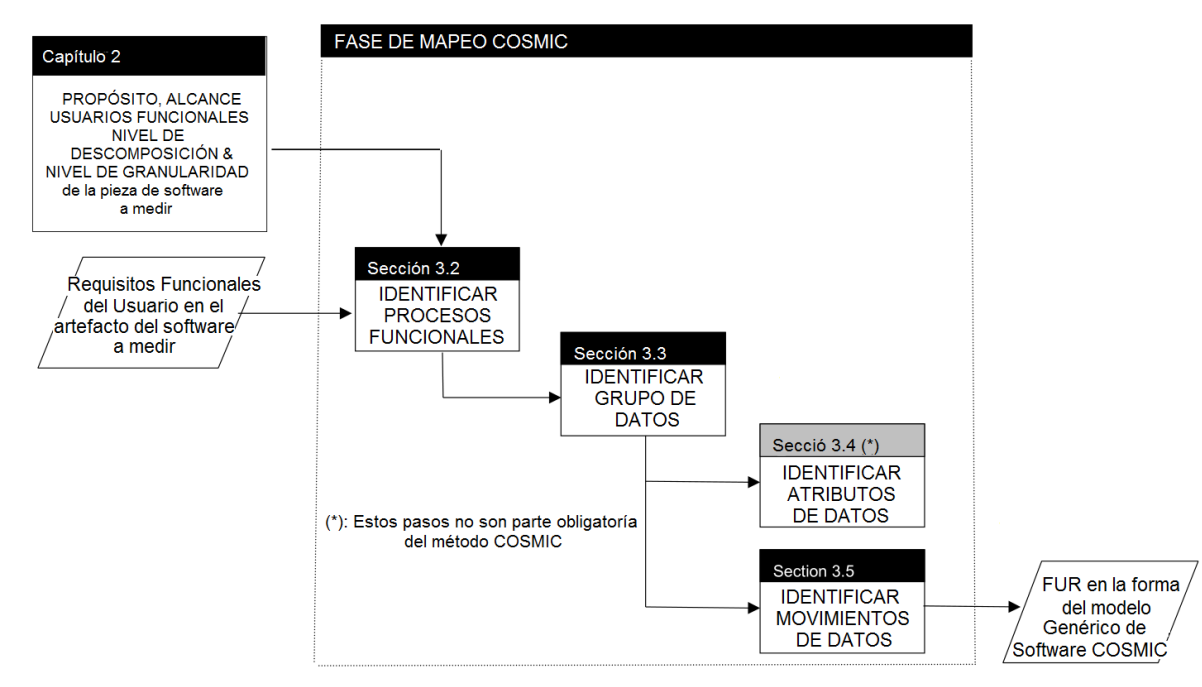


Figura 3.0 - Método general del proceso de mapeo COSMIC

Hay varias pautas disponibles que describen cómo mapear desde varios métodos de análisis de datos y determinación de requisitos, utilizados en diferentes dominios a los conceptos del método COSMIC. Algunos ejemplos son la Guía para dimensionar el software de aplicación empresarial [7], la Guía para dimensionar el software de aplicación de Data Warehouse [8], la Guía para dimensionar el software de arquitectura orientada a servicios [9] y la Guía para dimensionar el software en tiempo real [4]. Para los dominios empresariales [10] y en tiempo real [11], también hay disponibles Guías de referencia rápida que ofrecen 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 utilizado en la fase de Estrategia de medición al Modelo Genérico de Software. Las Figuras se aplican a una pieza de software de aplicación empresarial y a una pieza típica de software embebido en tiempo real, y corresponden a las Figuras 2.6 y 2.7 respectivamente.

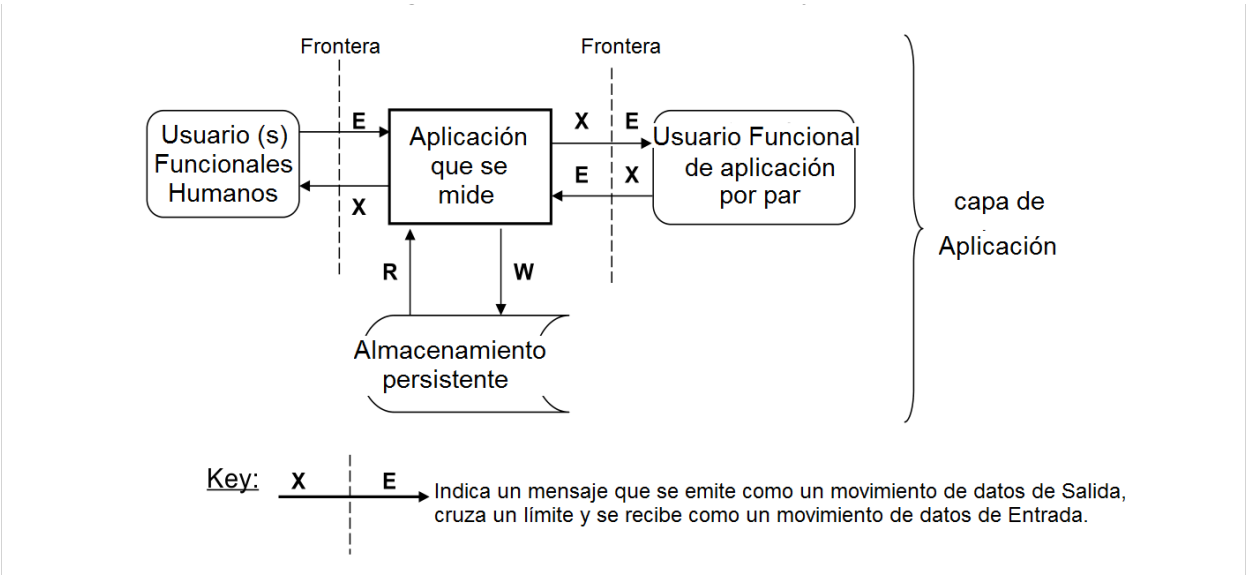


Figura 3.1 - Una aplicación de negocios con humanos y otra aplicación "igual" como usuarios funcionales

Por el principio f) del Modelo Contextual de Software (ver sección 1.3.1), observamos que los usuarios funcionales del software que se está midiendo son los "remitentes y/o destinatarios de 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 un límite/frontera y con un almacenamiento persistente dentro de este límite".

La figura 3.1 muestra que el software de aplicación que se está midiendo tiene dos usuarios funcionales, humano (s) y otra aplicación similar. La aplicación de software integrada de la Figura 3.2 solo tiene dispositivos de hardware y una aplicación similar como usuarios funcionales. Las flechas que muestran los movimientos de los datos están etiquetadas con las abreviaturas que muestran sus tipos (E = Entrada, X = Salida, R = Lectura, W = Escritura).

Tenga en cuenta que "almacenamiento persistente" se refiere a cualquier almacenamiento lógico al que el software que se está midiendo debe acceder mediante movimientos de datos de Lectura o Escritura; no implica ningún tipo de almacenamiento físico.

3.2 Identificando procesos funcionales.

El primer paso de la fase de medición es identificar el conjunto de procesos funcionales de la pieza de software a medir, a partir de los requisitos funcionales del usuario.

3.2.1 Definiciones

DEFINICIÓN – Evento.
Algo que sucede.

DEFINICIÓN – Evento desencadenante.
Un evento, reconocido en los Requisitos Funcionales del usuario del software que se mide, que hace que uno o más usuarios funcionales de este software generen cada uno o más grupos de datos. El primer grupo de datos generado por cualquier usuario funcional se moverá posteriormente mediante una Entrada desencadenante. Un evento desencadenante no puede ser subdividido y ha ocurrido o no ha ocurrido. NOTA: Los eventos de reloj y temporización pueden ser eventos desencadenantes.

DEFINICIÓN – Procesos funcionales.
a) Un conjunto de movimientos de datos, que representan una parte elemental de los Requisitos Funcionales del Usuario para el software que se mide, que es único dentro de estos FUR y que se puede definir independientemente de cualquier otro proceso funcional en estos FUR. b) Un proceso funcional tendrá una sola Entrada desencadenante. Cada proceso funcional comienza a procesarse cuando se recibe un grupo de datos movido por el movimiento de datos de Entrada desencadenante del proceso funcional. c) El conjunto de todos los movimientos de datos de un proceso funcional es el conjunto que se necesita para cumplir con sus FUR para todas las respuestas posibles a su Entrada desencadenante. NOTA 1: Cuando se implementa, es una *ocurrencia* de un proceso funcional que comienza a *ejecutarse* al recibir 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 adicionales además de la Entrada de desencadenante.

NOTA 3: Si un usuario funcional envía un grupo de datos con errores, por ejemplo, debido a que un sensor-usuario está funcionando mal o los datos ingresados por un ser humano tienen errores, generalmente es tarea del proceso funcional determinar si el evento realmente ocurrió y/o si los datos ingresados son realmente válidos, y cómo responder.

NOTA 4: Un proceso funcional es 'único' (como en a) arriba), y su tamaño total debe incluirse en el tamaño del FUR, si es iniciado por una Entrada desencadenante que resulta originalmente de un evento desencadenante que se distingue como único dentro del FUR. Dos o más procesos funcionales dentro del mismo FUR pueden ser únicos, incluso si comparten alguna funcionalidad común. Consulte la sección 3.2.7 para ver ejemplos de procesos funcionales con funcionalidad compartida.

DEFINICIÓN – Entrada Desencadenante.

El movimiento de datos Entrada de un proceso funcional que mueve un grupo de datos generado por un usuario funcional que el proceso funcional necesita para comenzar a procesar.

NOTA: Esta definición resulta del Modelo Genérico de Software, que es un modelo lógico. Físicamente, un proceso funcional puede comenzar a procesarse antes de que se hayan ingresado los datos, por ejemplo. cuando un usuario humano hace clic en un menú para mostrar una pantalla vacía para la entrada de datos.

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 ilustra a continuación en la Figura 3.3. La interpretación de este diagrama es: *un evento desencadenante hace que un usuario funcional genere un grupo de datos que es movido por la 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 decimos que un usuario funcional *inicia* una *Entrada desencadenante* que inicia un proceso funcional, o incluso más simplemente que un usuario funcional *inicia* un proceso funcional.

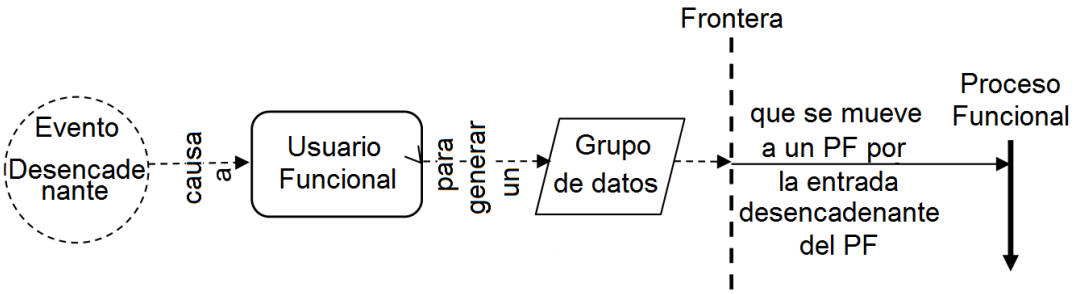


Figura 3.3 - Relaciones entre un evento desencadenante, un usuario y un proceso funcionales.

Todas las relaciones entre los conceptos de la Figura 3.3 (el evento desencadenante/usuario funcional / grupo de datos / Entrada desencadenante / proceso funcional) pueden ser de uno a muchos, muchos a uno o muchos a muchos, con una excepción. La excepción es que el grupo de datos movido por una Entrada desencadenante puede iniciar solo un proceso funcional según la cláusula b) de la definición de 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 que el evento debe ser notificado a la aplicación, es decir, es un evento desencadenante, el usuario inicia una Entrada desencadenante. (Un usuario humano puede incluso "generar" efectivamente un evento cuando decide realizar una consulta). De manera similar, una aplicación de software A que es un usuario funcional de otra aplicación B (que se está midiendo), puede "llamar" a la aplicación B para varios propósitos diferentes. Cada llamada (tipo) corresponde a la aplicación A que genera un evento de desencadenante por separado.
- Un usuario funcional hardware puede iniciar más de una Entrada desencadenante al detectar un evento de desencadenante en ciertos tipos de software críticos para la seguridad;
- Algunos procesos funcionales pueden ser iniciados por diferentes usuarios funcionales, por ejemplo, un componente de software reutilizable que puede ser llamado por cualquiera de sus usuarios funcionales de software.

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 hacen que usuarios funcionales específicos inicien procesos funcionales específicos) estarán limitadas 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.

Los grupos de datos que un usuario funcional generará como resultado de un evento desencadenante dependen del FUR del proceso funcional que procesará los datos, como se ilustra en los siguientes ejemplos.

EJEMPLO DE NEGOCIO 1: Un proceso funcional de un sistema de software de personal puede iniciarse mediante la Entrada desencadenante que mueve un grupo de datos que describe a un nuevo empleado. El grupo de datos es generado por un usuario funcional humano del software del personal que ingresa los datos.

EJEMPLO EN TIEMPO REAL 2: Un proceso funcional de un sistema de software en tiempo real puede iniciarse mediante su Entrada desencadenante que informa al proceso funcional que un reloj (usuario funcional) ha marcado. El grupo de datos que se movió transmite datos (la marca de verificación, tal vez a través de un solo bit) que solo informa que se ha producido un evento.

EJEMPLO EN TIEMPO REAL 3: Un proceso funcional de un sistema de software industrial de detección de incendios en tiempo real puede iniciarse mediante 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' (ocurrió un evento) e incluye la identificación del detector (es decir, los datos que se pueden usar para determinar dónde ocurrió el evento).

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

3.2.2 La forma de identificar los procesos funcionales

La forma de identificar procesos funcionales depende de los artefactos de software que están disponibles para el Medidor, que a su vez dependen del punto en el ciclo de vida del software en el que se requiere la medición y de los métodos de análisis, diseño y desarrollo de software en uso. Como las últimas varían enormemente, este Manual de Medición solo puede proporcionar un proceso general para identificar procesos funcionales.

El proceso para la identificación de procesos funcionales, una vez que se han identificado los usuarios funcionales, dado el FUR para el software que se está midiendo y considerando los ejemplos en el Apéndice C, sigue la cadena de la Figura 3.3, a saber:

- Identifique los eventos separados en el mundo de los usuarios funcionales a los que debe responder el software que se está midiendo, los "eventos desencadenantes" (los eventos desencadenantes pueden identificarse en los diagramas de estado y en los diagramas de ciclo de vida de la entidad, ya que algunas transiciones de estado y ciclo de vida de las transiciones de la entidad corresponden a eventos desencadenantes a los que el software debe reaccionar);
- Identifique qué usuario (s) funcional (es) del software puede responder a cada evento desencadenante;

- Identifique la Entrada desencadenante (o Entradas) que cada usuario funcional puede iniciar en respuesta al evento;
- Identificar el proceso funcional iniciado por cada Entrada desencadenante.

Utilice las siguientes reglas para verificar que los procesos funcionales candidatos se hayan identificado correctamente.

REGLAS – Procesos Funcionales.
a) Un proceso funcional pertenecerá íntegramente al alcance de la medición de una pieza de software en una, y sólo una, capa.
b) Un proceso funcional debe comprender un mínimo de dos movimientos de datos, es decir, la Entrada desencadenante más una Salida o una Escritura, dando un tamaño mínimo de 2 CFP. No hay límite superior para el número de movimientos de datos en un proceso funcional y, por lo tanto, no hay límite superior para su tamaño.
c) Un proceso funcional en ejecución se considerará terminado cuando haya cumplido con su FUR para todas las respuestas posibles a su Entrada desencadenante. Una pausa durante el procesamiento por razones técnicas no se considerará como finalización del proceso funcional.

3.2.3 Eventos desencadenantes y procesos funcionales en el dominio de aplicaciones de negocios.

a) Los eventos desencadenantes de una aplicación de negocios en línea usualmente ocurren en el mundo real de los usuarios funcionales humanos de la aplicación. El usuario humano transmite la ocurrencia del evento a un proceso funcional al ingresar datos sobre el evento.

   EJEMPLO DE NEGOCIO 1: En una empresa, se recibe un pedido (evento desencadenante), lo que hace que un empleado (usuario funcional) ingrese los datos del pedido (Entrada desencadenante que transmite datos sobre el objeto de interés 'pedido'), como el primer movimiento de datos del proceso funcional de 'entrada de pedidos'.

b) El evento desencadenante separado (-tipos) y, por lo tanto, el proceso funcional separado (-tipos) deben distinguirse en los siguientes casos:

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

   EJEMPLO DE NEGOCIO 2: Un usuario funcional ingresa un pedido de un cliente para un artículo de equipo industrial complejo y luego confirma la aceptación del pedido al cliente. Entre el ingreso del pedido y su aceptación, el usuario puede realizar consultas sobre si el nuevo pedido se puede entregar en la fecha de entrega solicitada y sobre la solvencia del cliente, etc. Aunque la aceptación de un pedido debe seguir el ingreso de un pedido, en este caso, el usuario debe tomar una decisión por separado para aceptar el pedido. Esto indica procesos funcionales separados para la entrada de pedidos y para la aceptación de pedidos (y para cada una de las consultas).

   Cuando las responsabilidades de las actividades están separadas.

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

c) Una aplicación A tiene una aplicación B que necesita enviar datos a, u obtener datos de, la aplicación A. La aplicación B inicia un proceso funcional en la aplicación A cuando necesita enviar datos u obtener datos de la aplicación A. La aplicación B es entonces Un usuario funcional de la aplicación A.

   EJEMPLO DE NEGOCIO 4: Supongamos que al recibir un pedido en el Ejemplo de negocio 1, la aplicación de procesamiento de pedidos debe enviar los detalles de cualquier cliente nuevo a una aplicación de registro de clientes central, que se está midiendo. La aplicación de procesamiento de pedidos es, por tanto, un usuario funcional de la aplicación central. La aplicación de procesamiento de pedidos, después de haber

recibido datos sobre un cliente nuevo, genera el grupo de datos del cliente que envía a la aplicación central de registro de clientes que activa un proceso funcional para almacenar estos datos.

d) No hay diferencia en principio para el análisis de un proceso funcional si se requiere que se procese en línea o en modo de proceso por lotes. Por definición, todos los datos de entrada para el procesamiento por lotes deben haberse almacenado temporalmente en algún lugar antes de que pueda comenzar el proceso, o pueden transmitirse como un flujo por lotes. Ver Figura 3.4. (NB: distinguimos los datos de entrada, todas las Entradas, de los datos persistentes que también pueden necesitar ser leídos o escritos por el proceso por lotes). Al medir el software de procesamiento por lotes, los datos de entrada almacenados temporalmente o el flujo de datos de entrada deben analizarse en el de la misma manera que si se ingresara directamente a la aplicación. Estos datos de entrada no son "datos persistentes".

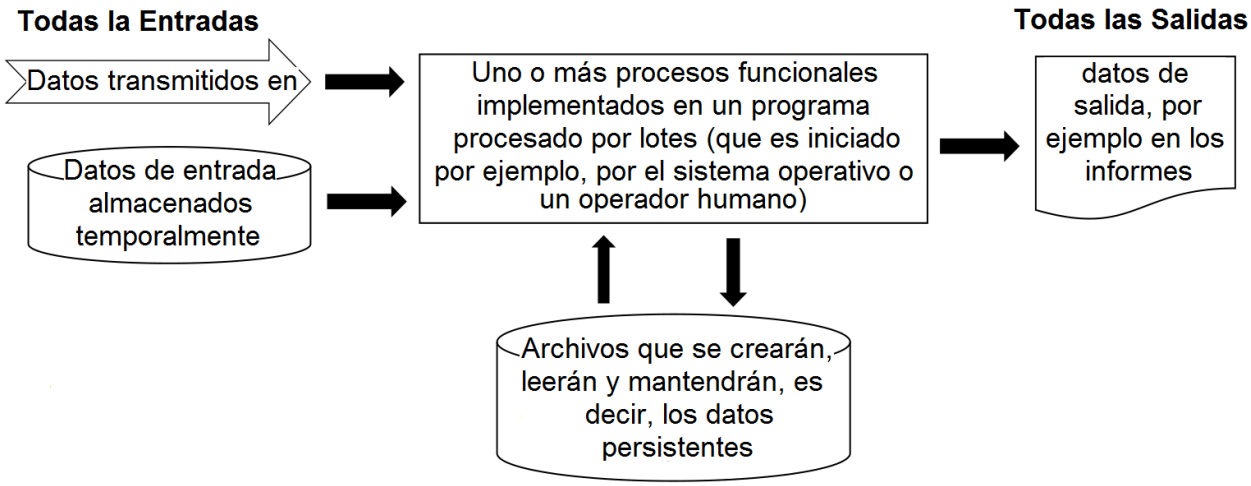


Figura 3.4 - Un 'trabajo' procesado por lotes, que implementa una colección de procesos funcionales

NOTA: Un requisito en donde algunos datos de entrada se procesen por lotes es un requisito no funcional (NFR). El efecto de este NFR es que los datos de entrada deben estar disponibles (como un "lote") para ingresar a la aplicación procesada por lotes. La forma en que esto sucede en la práctica no concierne al análisis del FUR del software procesado por lotes.

Tenga en cuenta también que cada proceso funcional (tipo) que se va a procesar en un "trabajo" por lotes debe analizarse 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.

EJEMPLO DE NEGOCIO 5: Supongamos que los pedidos en el Ejemplo de Negocio 1 anterior se ingresan de alguna manera mediante un proceso "fuera de línea", por ejemplo, mediante escaneo óptico de documentos en papel y se almacenan temporalmente para el procesamiento automático de lotes. ¿Cómo se analiza este proceso funcional de entrada de pedidos si se va a procesar por lotes? El usuario funcional es el humano que hace que los datos del pedido se ingresen fuera de línea listos para ser procesados en modo batch; la Entrada desencadenante del proceso funcional que procesará los pedidos en modo de proceso por lotes es el movimiento de datos que mueve el grupo de datos de pedidos desde el almacenamiento temporal al proceso. (El proceso fuera de línea, si debe medirse, implica otro proceso funcional separado. Efectivamente, el usuario funcional ha iniciado dos Entradas desencadenantes, una inicia el proceso fuera de línea para cargar las órdenes en el almacenamiento temporal y la otra para comenzar El procesamiento por lotes de los pedidos.)

EJEMPLO DE NEGOCIO 6: Supongamos que un FUR para una aplicación de procesamiento por lotes de fin de año es informar el resultado del negocio para el año y restablecer las posiciones para el comienzo del próximo año. Físicamente, una marca de reloj de fin de año generada por el sistema operativo hace que la aplicación comience a procesarse. Sin embargo, lógicamente, cada proceso funcional de la aplicación toma sus datos de entrada del flujo de datos para ser procesados por lotes. Esto debe analizarse de la manera normal (por ejemplo, los datos de entrada para cualquier proceso funcional comprenden una o más Entradas, la primera de las cuales es la Entrada desencadenante para ese proceso)

Sin embargo, supongamos que hay un proceso funcional particular de la aplicación de procesamiento por lotes que no requiere ningún dato de entrada para producir su conjunto de informes. Físicamente, el usuario funcional (humano) ha delegado en el sistema operativo la tarea de desencadenar este proceso funcional. Dado que cada proceso funcional debe tener una Entrada desencadenante, podemos considerar que la marca del reloj de fin de año que inició el flujo de lotes cumple este rol en este proceso. Este proceso funcional puede necesitar varias 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 a través de un clic del ratón en un elemento de menú en línea, en lugar de delegar la activación de la producción del informe en modo de lotes al sistema operativo.

Para ver varios ejemplos sobre cómo distinguir eventos desencadenantes y procesos funcionales en flujos por lotes, consulte la 'Guía para dimensionar el software de aplicación de negocios usando COSMIC' [7], sección 4.6.3.

3.2.4 Eventos desencadenantes y procesos funcionales en el dominio de aplicaciones en tiempo real

a) Un sensor suele detectar un evento desencadenante.

EJEMPLO EN TIEMPO REAL 1: Cuando un sensor (usuario funcional) detecta que la temperatura alcanza un cierto valor (evento desencadenante), el sensor envía una señal para iniciar un movimiento de datos de entrada de un proceso funcional para apagar un calentador (otro usuario funcional).

EJEMPLO EN TIEMPO REAL 2: Un avión militar tiene un sensor que detecta el evento "que se acerca un misil". El sensor es un usuario funcional del software que debe responder a la amenaza. Para este software, se produce un evento solo cuando el sensor detecta algo, y es el sensor (el usuario funcional) el que genera un grupo de datos para iniciar la entrada desencadenante dicha, por ejemplo. El "sensor 2 ha detectado un misil", además de un flujo de datos acerca de qué tan rápido se aproxima el misil y sus coordenadas.

b) Las señales periódicas de un reloj ("tics de reloj") pueden desencadenar un proceso funcional.

Vea el Ejemplo 2 en tiempo real en la sección 3.2.1. No hay otros datos que acompañen al reloj. El proceso funcional obtiene datos de varios sensores y toma las medidas necesarias.

3.2.5 Más sobre procesos funcionales separados

El software distingue los eventos y proporciona los procesos funcionales correspondientes en función únicamente de su FUR. Al dimensionar el software, a veces puede ser difícil decidir cuáles son los eventos separados que el software debe reconocer. Este es especialmente el caso en el que los requisitos originales ya no están disponibles o cuando, por ejemplo, el desarrollador puede haber encontrado económico implementar varios requisitos en una transacción física. Puede ayudar con el análisis examinar la organización de la entrada de datos (ver más 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 procesos funcionales correspondientes.

EJEMPLO DE NEGOCIO 1: Supongamos que existe un requisito funcional de usuario para dos tipos de beneficios sociales, primero para un hijo adicional y segundo un "crédito fiscal de trabajo" para aquellos con bajos ingresos. Estos son requisitos para que el software responda a dos eventos que están separados en el mundo de los usuarios funcionales humanos. Por lo tanto, debe haber dos procesos funcionales, aunque se haya utilizado un formulario de impuestos único para capturar datos para ambos casos.

De acuerdo con la cláusula c) de su definición, un proceso funcional debe "cumplir con su FUR para todas las respuestas posibles a su Entrada desencadenante". Esto significa que el mismo tipo de proceso funcional debe ser capaz de tratar todas *las posibles ocurrencias de valores* de los atributos de datos del grupo de datos movido por su Entrada desencadenante, incluidos valores de datos válidos e inválidos, e incluso en algunos casos valores de datos perdidos. Todas estas variaciones en los valores de los datos movidos por la Entrada desencadenante generalmente darán lugar a que se sigan diferentes rutas de procesamiento cuando se ejecute el proceso funcional. Pero a pesar de todas estas variaciones, todavía debemos identificar un solo tipo de proceso funcional iniciado por su único tipo de Entrada desencadenante. (El Medidor solo 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).

EJEMPLO DE NEGOCIO 2: Un proceso funcional que proporciona una capacidad de búsqueda general contra una base de datos puede ser requerido para aceptar hasta cuatro parámetros de búsqueda (atributos de su Entrada desencadenante). Pero el mismo proceso funcional funcionará si se ingresan los valores de solo uno, dos o tres parámetros de búsqueda

EJEMPLO DE NEGOCIO 3: Para que un proceso funcional registre a un nuevo cliente para una empresa de alquiler de autos, es obligatorio ingresar datos para la mayoría de los atributos de datos, pero algunos (por ejemplo, algunos detalles de contacto) son opcionales y pueden dejarse vacíos. Independientemente de si se ingresan todos o un subconjunto de estos atributos, solo hay un proceso funcional para registrar un nuevo cliente.

EJEMPLO DE NEGOCIO 4: Continuando con el Ejemplo 3, para el proceso funcional de hacer una reserva de alquiler de automóvil en la misma compañía, hay varias opciones que pueden o no ser aceptadas, por ejemplo. para seguros adicionales, conductores adicionales, solicitudes de asientos para niños, etc. Estas diferentes opciones conducen a diferentes rutas de procesamiento dentro del proceso funcional de reserva de alquiler de autos, pero todavía hay un solo proceso funcional para reservar un alquiler de autos.

EJEMPLO EN TIEMPO REAL: Una entrada desencadenante (información de altitud del avión enviada por el Sistema de posicionamiento geográfico) a un proceso funcional de un sistema de aviónica desencadenará una de dos rutas de procesamiento bastante diferentes dentro del proceso funcional, dependiendo del valor de la Entrada, es decir, si la altitud está por encima o por debajo de una altura dada. Las diferentes rutas mostrarán diferentes grupos de datos en el mapa del piloto y, si la altitud es demasiado baja, se emitirán advertencias adicionales. Solo hay un proceso funcional.

Una vez identificado, cada proceso funcional puede registrarse en una línea individual, bajo la capa apropiada y la pieza de software nombrada, en la matriz del Modelo genérico de software (Apéndice A).

3.2.6 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, se debe definir un alcance de medición separado para cada componente. En tal caso, el tamaño de los procesos funcionales de cada componente sigue todas las reglas normales como ya se ha descrito.

Del proceso para cada medición (... defina el alcance, luego los usuarios funcionales y la frontera, etc.) se deduce que, si una pieza de software consta de dos o más componentes, no puede haber superposición entre el alcance de medición de cada componente. El alcance de medición para cada componente debe definir un conjunto de procesos funcionales completos. Por ejemplo, no puede haber un proceso funcional con parte en un alcance y parte en otro. Del mismo modo, los procesos funcionales dentro del alcance de medición para un componente no tienen información sobre los procesos funcionales dentro del alcance de otro componente, incluso aunque los dos componentes intercambien mensajes.

El (los) usuario (s) funcional (es) de cada componente se determina (n) al examinar dónde ocurren los eventos que desencadenan procesos funcionales en el componente que se está examinando. (Los eventos desencadenantes solo 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 cliente-servidor y los movimientos de datos que intercambian.

3.2.7 Independencia de los procesos funcionales que comparten alguna funcionalidad común o similar: reutilización.

Cualquiera de dos o más procesos funcionales en el mismo software que se mide puede tener alguna funcionalidad que sea idéntica o muy similar en cada proceso. Este fenómeno se conoce como "funcionalidad en común" o "similitud" funcional.

Sin embargo, en el método COSMIC (como en todos los demás métodos FSM), cada proceso funcional se define, modela y mide independientemente, es decir, sin referencia a, cualquier otro proceso funcional en el mismo software que se esté midiendo (consulte la cláusula a) de la definición del proceso funcional). Cualquier funcionalidad requerida que sea común o similar en cualquiera de los dos o más procesos funcionales en el mismo software que se está midiendo se debe incluir al medir el tamaño de cada uno de estos procesos funcionales. Los siguientes son ejemplos de funcionalidad en común o similitudes funcionales que se pueden encontrar en la práctica.

EJEMPLO DE NEGOCIO: Varios procesos funcionales en el mismo software que se mide pueden necesitar la misma funcionalidad de validación, por ejemplo. "Fecha de pedido", o puede que necesite acceder a los mismos datos persistentes, o que deba realizar el mismo cálculo de intereses.

EJEMPLO EN TIEMPO REAL: Varios procesos funcionales en el mismo software que se está midiendo pueden necesitar obtener datos del mismo sensor (movimiento común del mismo grupo de datos) o pueden requerir el mismo cálculo de conversión de escala, por ejemplo. de Fahrenheit a Centígrado (manipulación de datos común).

Cuando se implementa una declaración de FUR en el software, cualquier "funcionalidad en común" puede o no desarrollarse como software reutilizable. Todas las decisiones de implementación, incluido el alcance de la reutilización real o potencial del software, deben ignorarse al medir el tamaño funcional. Sin embargo, es posible que sea necesario tener en cuenta la reutilización al usar mediciones de tamaño funcional para el análisis del esfuerzo del proyecto o propósito de estimación.

3.2.8 Eventos que desencadenan un sistema de software para comenzar a ejecutarse

Al medir el tamaño de una pieza de software, identifique solo los eventos y las correspondientes Entradas desencadenantes que desencadenan los procesos funcionales a los que el software debe responder según lo definido en su FUR. La funcionalidad necesaria para iniciar (o "disparar") el software en sí no forma parte de estos procesos funcionales y debe ignorarse (o medirse por separado, si es necesario). Los siguientes ejemplos describen cómo se inicia el software en tres dominios.

EJEMPLO DE NEGOCIO: para una aplicación de negocios, el usuario funcional que inicia la aplicación puede ser un componente programador del sistema operativo, un operador de computadora o cualquier otro usuario humano (por ejemplo, cuando un usuario de PC inicia un navegador o un software de procesamiento de texto).

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

EJEMPLO DE INFRAESTRUCTURA: Para un sistema operativo de computadora, el usuario funcional que inicia el sistema operativo es un programa de arranque que se inicia cuando se enciende la computadora.

Los siguientes son ejemplos de dos dominios de las relaciones que pueden existir, si las hay, entre el evento/proceso que inicia el software que se va a medir y los eventos/procesos que el software debe ejecutar como se describe en su FUR.

EJEMPLO DE NEGOCIO 1: Un calendario del sistema operativo puede iniciar una aplicación para procesar los datos de entrada para una variedad de procesos funcionales en modo de proceso por lotes. Si el propósito es medir el FUR de la aplicación por lotes, se debe ignorar la funcionalidad de "inicio del sistema". Las Entradas desencadenantes para los procesos funcionales de la aplicación procesada por lotes y cualquier otra Entrada que pueda ser necesaria formarán los datos de entrada para la aplicación por lotes.

EJEMPLO DE NEGOCIO 2: Excepcionalmente, una aplicación procesada por lotes para producir informes de resumen al final de un período de tiempo puede iniciarse sin necesidad de datos de entrada proporcionados directamente por el usuario funcional. Para el análisis, ver Ejemplo de Negocio 6 en la sección 3.2.3.

EJEMPLO EN TIEMPO REAL: Un vehículo moderno tiene un sistema distribuido de Unidades de Control Electrónico (ECU) para controlar muchas funciones, por ejemplo. administración del motor, frenos, aire acondicionado, etc. En la arquitectura AUTOSAR, en un sistema distribuido, el módulo de 'Administración de red' (NM), que siempre está en ejecución, es responsable de activar las ECU que están conectadas entre sí a través de una red ('bus'). Este módulo NM también maneja la conmutación coordinada entre los estados operativos de la ECU: Operación Normal, Baja Potencia y Reposo. Por lo tanto, es el NM el que se despierta o pone en suspensión las ECU. Al medir cualquier software de aplicación de ECU, esta funcionalidad NM debe ignorarse.

3.3 Identificación de 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 identificar sus movimientos de datos. Para hacer esto, debemos recordar el Modelo Genérico de Software (consulte la sección 1.3.2) y la introducción a los conceptos de este capítulo en la sección 3.0, que establecen que un movimiento de datos

'mueve un grupo de datos, cuyos atributos de datos describen un solo objeto de interés'. Por lo tanto, para comprender un movimiento de datos, primero debemos definir y entender estos tres conceptos.

DEFINICION – Objeto de interés.
Cualquier "cosa" en el mundo del usuario funcional que se identifica en los Requisitos Funcionales del Usuario del software que se mide, sobre el cual se requiere que el software mueva un grupo de datos dentro o fuera del software, o hacia o desde el almacenamiento persistente. Puede ser cualquier cosa física, así como cualquier objeto conceptual o parte de un objeto conceptual. NOTA 1: En el método COSMIC, el término "objeto de interés" se utiliza para evitar términos relacionados con métodos específicos de ingeniería de software. El término no implica "objetos" en el sentido utilizado en los métodos orientados a objetos. NOTA 2: Cuando un usuario funcional envía un grupo de datos que se describe a sí mismo, por ejemplo, su estado o identidad, o cuando un usuario funcional recibe datos que se describen a sí mismos, entonces el usuario funcional también cumple el rol de la "cosa" en su mundo, por lo que también es el objeto de interés descrito por el grupo de datos. NOTA 3: No hay nada absoluto sobre un objeto de interés. Una "cosa" puede ser un objeto "de interés" para un usuario funcional a través de uno o más procesos funcionales, pero no puede ser un objeto "de interés" para otro usuario funcional a través de otros procesos funcionales, incluso en el mismo software que se mide.

Para más información sobre la Nota 2 de esta definición, consulte la sección 3.3.4.

DEFICIÓN – Grupo de datos.
Un conjunto de atributos de datos distinto, no vacío y no ordenado donde cada atributo de dato incluido describe un aspecto complementario del mismo objeto de interés. NOTA: el término "grupo de datos" no significa necesariamente "el conjunto de *todos* los atributos de datos que describen un único objeto de interés". El FUR de una parte del software puede especificar que se formen grupos de datos a partir de cualquier combinación de atributos de datos que describan el mismo objeto de interés, según sea necesario por diferentes procesos funcionales

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

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

DEFINICIÓN – Atributo de dato.
Es la pieza de información más pequeña, dentro de un grupo de datos identificado, llevando un significado desde la perspectiva de los Requisitos del Usuario Funcional del Software. NOTA: Sinónimo 'Elemento de datos'

(Para obtener más información sobre los atributos de los datos, consulte la sección 3.4.)

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

a) Como una estructura de registro físico en un dispositivo de almacenamiento de hardware (archivo, tabla de base de datos, memoria ROM, etc.).

b) Como una estructura física dentro de la memoria volátil de la computadora (estructura de datos asignada de forma dinámica o mediante un bloque de espacio de memoria asignado previamente).

c) Como la presentación agrupada de atributos de datos relacionados funcionalmente en un dispositivo de entrada/salida (pantalla de visualización, informe impreso, pantalla de panel de control, etc.).

d) Como un mensaje en transmisión entre un dispositivo y una computadora, o a través de una red, etc.

Una vez identificado, cada grupo de datos se puede registrar en una columna individual en la matriz del Modelo Genérico de Software (Apéndice A), bajo el encabezado "Nombre del grupo de datos".

3.3.2 Sobre la identificación de objetos de interés y grupos de datos.

La definición y los principios de los objetos de interés y de los grupos de datos son intencionalmente amplios para ser aplicables a la gama más amplia posible de software. Esta calidad a veces hace que sea difícil aplicar la definición y los principios al medir una pieza específica de software. Por lo tanto, los siguientes ejemplos están diseñados para ayudar en la aplicación de los principios a casos específicos.

Cuando se enfrenta a la necesidad de analizar un grupo de atributos de datos que se mueven dentro o fuera de un proceso funcional o es movido por un proceso funcional hacia o desde el almacenamiento persistente, *es de vital importancia* decidir si todos los atributos transmiten datos sobre un solo 'objeto de interés', ya que es este último el que determina el número de 'grupos de datos' separados según lo definido por el método COSMIC que se moverá por los movimientos de datos. Por ejemplo, si los atributos de datos que se ingresarán en un proceso funcional son atributos de tres objetos de interés separados, entonces debemos identificar tres movimientos de datos de "Entrada" separados.

Este problema de decidir el número de grupos de datos puede ser especialmente difícil cuando se analiza el resultado de un proceso funcional de una aplicación empresarial que puede incluir:

- Múltiples grupos de datos, cada uno describe un objeto de interés diferente, por ejemplo, un informe que muestra los totales en varios niveles de agregación;
- los resultados de las consultas donde la salida variará dependiendo de la entrada;
- grupos de datos que pueden no estar relacionados entre sí, por ejemplo, una factura que incluye un anuncio de un servicio no relacionado.

La siguiente regla pretende ayudar a la identificación de grupos de datos y, por lo tanto, de objetos de interés, particularmente en la salida de procesos funcionales. Sin embargo, la regla se aplica igualmente a la entrada a un proceso funcional, así como a sus movimientos hacia y desde el almacenamiento persistente y al FUR del software en cualquier dominio funcional.

REGLA - Identificar diferentes grupos de datos (y, por lo tanto, diferentes objetos de interés) movidos en el mismo proceso funcional
Para todos los atributos de datos que aparecen en la entrada de un proceso funcional: a) los conjuntos de atributos de datos que tienen diferentes frecuencias de ocurrencia describen diferentes objetos de interés; b) conjuntos de atributos de datos que tienen la misma frecuencia de ocurrencias, pero diferentes atributos clave de identificación describen diferentes objetos de interés; c) todos los atributos de datos en un conjunto que resultan de aplicar las partes a) y b) de esta regla pertenecen al mismo tipo de grupo de datos, a menos que el FUR especifique que puede haber más de un tipo de grupo de datos que describa el mismo objeto de interés en la entrada al proceso funcional (ver Nota 3) Esta misma regla se aplica a todos los atributos de datos que aparecen en la salida de un proceso funcional, o todos los que se mueven de un proceso funcional a un almacenamiento persistente, o todos los que se mueven de un almacenamiento persistente a un proceso funcional. NOTA 1. Puede ser útil al analizar resultados complejos, por ejemplo, informes con datos que describen varios objetos de interés, para considerar cada grupo de datos candidato separado como si fuera un proceso funcional separado. Cada uno de los tipos de grupos de datos identificados de esta manera también debe distinguirse y contarse al medir el informe complejo.

Para ver ejemplos, consulte la 'Guía de Medición para las Aplicaciones de Negocios' [7], en particular el ejemplo en la sección 2.6.1 y su análisis en 2.6.2. Ver también el análisis de los ejemplos 4 y 5 en la sección 4.2.4.

NOTA 2. Examinar cómo los atributos de datos se agrupan o se separan físicamente en la *entrada o en la salida* puede ayudar a distinguir diferentes tipos de grupos de datos, pero no se puede confiar en ellos para distinguirlos. Como ejemplo, dos o más conjuntos de atributos de datos que ocurren en la misma entrada o salida que están físicamente separados por razones estéticas o para facilitar la comprensión, pertenecerán al mismo tipo de grupo de datos si cumplen con la regla anterior.

NOTA 3. Consulte la sección 3.5 del Manual de Medición para las definiciones, los principios y las reglas para los *movimientos de datos* que mueven grupos de datos, y la sección 3.5.7 (Ejemplos 2, 3, 4 y 5) y 3.5.11 para las excepciones a estas reglas para movimientos de datos, según la regla c anterior.

Vea el Ejemplo de Negocio 3 en la sección 3.3.2 como una ilustración simple de la aplicación de esta regla. Para obtener ejemplos más complejos, consulte la "Guía para la Medición Software de Aplicaciones de Negocios", [7].

Objetos de interés y grupos de datos en el dominio de aplicaciones de negocios.

En las aplicaciones de negocios, los datos son "persistentes" si permanecen almacenados más allá de la vida útil de un proceso funcional, o "transitorios" si existen solo en la entrada o salida de un proceso funcional.

Nota: la distinción entre persistente o transitorio no se aplica a los objetos de interés; son "cosas", físicas o conceptuales, identificadas en el FUR del software a medir.

EJEMPLO DE NEGOCIO 1: En el dominio del software de aplicación de negocios, un objeto de interés podría ser "empleado" (físico) o "orden" (conceptual). Los grupos de datos que describen que estos objetos de interés se harían persistentes si se requiere que el software almacene datos sobre empleados o pedidos. En el caso de "orden", comúnmente se deduce del FUR de los pedidos de varias líneas que se identifican dos objetos de interés: "orden" y "línea de orden". Los grupos de datos correspondientes podrían denominarse "datos de orden" y "datos de línea de orden".

Los grupos de datos transitorios se forman siempre que hay una consulta (o solicitud de informe) que solicita datos sobre una o más 'cosas', es decir, 'objetos de interés', sobre qué datos no se guardan en el almacenamiento persistente, pero que pueden derivarse a partir de datos almacenados en almacenamiento persistente o derivados simplemente por la manipulación de datos. El grupo de datos de entrada de dicha consulta (los parámetros de selección para obtener los datos requeridos) y el grupo o grupos de datos de salida (que contienen los atributos deseados) son grupos de datos transitorios porque existen solo en la entrada y en la salida, respectivamente, y no sobrevivir a la ejecución del proceso funcional.

EJEMPLO DE NEGOCIO 2: Suponga un FUR para un proceso funcional de consulta ad hoc en una base de datos de personal para averiguar el número de empleados mayores de una edad determinada, que se deben ingresar. El parámetro de entrada (el límite de edad) es un grupo de datos transitorios que define un objeto de interés "el conjunto de empleados que superan el límite dado". El grupo de datos de salida, que comprende el recuento de empleados de edad por encima del límite dado, también es transitorio y describe el mismo objeto de interés que la entrada. Los datos de los que se deriva el recuento (el archivo del empleado) son persistentes.

EJEMPLO DE NEGOCIO 3: Suponga la misma consulta ad hoc contra una base de datos de personal como en el Ejemplo de negocio 2, pero además de dar salida al número total de empleados por encima del límite de edad dado, la consulta también debe enumerar los nombres de todos los empleados de edad superior al límite dado. El análisis es como en el ejemplo de negocio 2, pero el proceso funcional ahora debe generar dos grupos de datos: el recuento de empleados (datos transitorios) y la lista de nombres de empleados sobre el límite de edad dado (derivado de datos persistentes). Estos deben ser dos grupos de datos separados porque tienen diferentes frecuencias de ocurrencia (un recuento de empleados para cero, uno o más nombres de empleados (ver la regla anterior),

Una lección general importante del Ejemplo de Negocio 3 es que un "conjunto" y un "miembro de un conjunto" son siempre "cosas" diferentes, por lo que deben ser objetos de interés diferentes. En el caso de este ejemplo, el proceso funcional genera grupos de datos que describen dos objetos de interés, a saber, el "conjunto de

empleados más antiguo que el límite de edad dado" y "empleado". Sus grupos de datos de atributo único son "recuento de empleados por encima del límite de edad" y "nombre del empleado", respectivamente.

Para una discusión detallada sobre los métodos de análisis de datos para determinar objetos de interés y grupos de datos separados, se remite al lector a la 'Guía para la Medición Software de Aplicaciones de Negocios que utiliza COSMIC' [7].

Objetos de interés y grupos de datos en el dominio de software en tiempo real.

En el software en tiempo real, generalmente no hay necesidad de pensar en los datos como transitorios. En general, todos los datos que se mueven dentro o fuera de un proceso funcional en tiempo real son transitorios a menos que se hagan persistentes (por ejemplo, para fines de registro), o se hayan obtenido del almacenamiento persistente. Los siguientes son ejemplos de grupos de datos y objetos correspondientes de interés en software en tiempo real.

EJEMPLO 4 EN TIEMPO REAL: Un grupo de datos que ingresa software desde un dispositivo físico puede informar:

- *sobre el estado actual de un objeto de interés, como que una válvula está abierta o cerrada. (Consulte la sección 3.3.4 sobre cuándo un usuario funcional envía datos sobre sí mismo, por lo que también es objeto de interés).*
- *que ha ocurrido un evento desencadenante, lo que lleva al inicio de un proceso funcional (que puede interrumpir un proceso funcional que ya se está ejecutando). El evento es el objeto de interés.*

De manera similar, una salida de grupo de datos a un dispositivo físico, como encender o apagar una lámpara de advertencia, transmite datos sobre el objeto de interés de la lámpara.

EJEMPLO DE TIEMPO REAL 5: Un sistema de software de conmutación de mensajes puede recibir un grupo de datos de mensajes como entrada y enrutarlo hacia adelante sin cambios como salida, según la FUR de la pieza particular del software. Los atributos del grupo de datos de mensaje podrían ser, por ejemplo, 'ID de mensaje, ID de remitente, ID de destinatario, código de ruta y mensaje contenido'; el objeto de interés del mensaje es "mensaje".

EJEMPLO 6 EN TIEMPO REAL: Una estructura de datos común representa objetos de interés que se mencionan en el FUR, que pueden mantenerse mediante procesos funcionales y que es accesible para la mayoría de los procesos funcionales que se encuentran en el software medido.

EJEMPLO DE TIEMPO REAL 7: Una estructura de datos de referencia, representa objetos de interés cuyos valores de atributo se dan en gráficos o tablas que se encuentran en el FUR, y que se guardan en la memoria permanente (memoria ROM, por ejemplo) y son accesibles para la mayoría de los Procesos funcionales encontrados en el software medido.

EJEMPLO EN TIEMPO REAL: 8: Los archivos, comúnmente designados como "archivos planos", representan objetos de interés mencionados en el FUR, que se guardan en un dispositivo de almacenamiento persistente.

3.3.3 Datos o grupos de datos que no son candidatos para los movimientos de datos

Los datos que aparecen en las pantallas de entrada o salida o los informes que no están relacionados con un objeto de interés para un usuario funcional no deben identificarse como indicadores de un grupo de datos, por lo que no deben medirse.

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

EJEMPLO DE NEGOCIO 2: "Comandos de control" (un concepto definido solo en el dominio de la aplicación de negocios) que permite a un usuario funcional controlar su uso del software, en lugar de mover datos, por ejemplo. los comandos de avance/retroceso de página, haga clic en "Aceptar" para confirmar un mensaje de error, etc., consulte la sección 3.5.10.

El modelo genérico de Software de COSMIC asume que toda manipulación de datos dentro de un proceso funcional está asociada con los cuatro tipos de movimiento de datos, consulte la sección 3.5.6. Por lo tanto, no se pueden identificar grupos de datos que surjan de la manipulación de datos dentro de un proceso funcional además de los grupos de datos movidos por las Entradas, Salidas, Lecturas y Escrituras del proceso. (Para ver

ejemplos de manipulación y movimientos de datos que pueden malinterpretarse como movimientos de datos, consulte la sección 3.5.4, principio c) para una lectura, y la sección 3.5.5 principio d) para una escritura).

3.3.4 El usuario funcional como objeto de interés.

Como se señala en la NOTA 2 de la definición de un objeto de interés, un usuario funcional del software que se está midiendo también puede cumplir la función del objeto o el interés de un grupo de datos enviado o recibido por el usuario funcional.

En muchos sistemas de software en tiempo real, como se describe en el Ejemplo de tiempo real 1 en la sección 3.3.2 anterior, la fase de Estrategia puede mostrar que un dispositivo físico como un sensor podría ser un usuario funcional. Más adelante, en la fase de Mapeo, este mismo sensor también puede ser identificado como un objeto de interés. Esto se debe simplemente a que el dispositivo físico "interactúa con el software" y, al mismo tiempo, el dispositivo físico es una "cosa ... sobre la cual se requiere que el software mueva un grupo de datos dentro o fuera del software, o hacia o desde almacenamiento persistente'. En efecto, el dispositivo físico interactúa con el software a través del límite para proporcionar o recibir información sobre sí mismo en forma de un grupo de datos que describe algún aspecto de sí mismo.

De manera similar, en el software de aplicación empresarial, un usuario funcional humano también puede ser objeto de interés del grupo de datos introducido o recibido por el usuario humano.

> *EJEMPLO DE NEGOCIO: Un usuario funcional humano ingresa una identificación y una contraseña en un proceso de inicio de sesión para identificarse en un sistema. El objeto de interés del grupo de datos introducido es el usuario humano.*
>
> *EJEMPLO EN TIEMPO REAL: Supongamos que un sensor de temperatura 'A' envía una medida de la temperatura actual de un material para procesarlo mediante un proceso funcional. En una vista, se puede considerar que el sensor proporciona información sobre su propio estado. Por lo tanto, el sensor cumple con los criterios de identificación como un objeto de interés y puede ser mapeado desde el FUR como tal. Sin embargo, el sensor físico está en el mundo de los usuarios y '... interactúa con el software ... a través del límite', por lo tanto, también cumple con los criterios de identificación como usuario funcional y, por lo tanto, debe aparecer en el diagrama de contexto del software.*

3.4 Identificación de atributos de datos (opcional)

En esta sección se analiza la identificación de los atributos de datos a los que hace referencia el software a medir. En el método COSMIC, no es obligatorio identificar los atributos de los datos. Sin embargo, es necesario comprender el concepto de un "atributo de datos" para entender la sección sobre "medir cambios ...", donde un FUR para cambiar un atributo de datos puede dar como resultado el movimiento de datos al que pertenece el atributo que se está midiendo. Además, puede ser útil analizar e identificar atributos de datos en el proceso de distinguir grupos de datos y objetos de interés. Los atributos de datos también se pueden identificar si se requiere una subunidad de la medida del tamaño, como se presenta en la sección 4.5 "Ampliación del método de medición COSMIC".

3.4.1 Ejemplos de atributos de datos

(Para la definición de un atributo de datos, consulte la sección 3.3.1.)

> *EJEMPLO DE NEGOCIO 1: Un objeto de interés de 'empleado' puede ser descrito por un grupo de datos llamado 'Datos maestros de empleado', que contiene los atributos de datos 'ID de empleado', 'Nombre', 'Dirección', 'Fecha de nacimiento', ' Sexo ',' Estado civil ',' Número de seguro nacional ',' Grado ',' Título laboral', etc.*
>
> *EJEMPLOS DE TIEMPO REAL 2: Un par termoeléctrico puede, a petición, informar el atributo "Temperatura", Un sensor de un sistema de seguridad puede detectar un intruso y enviar el atributo "Movimiento detectado". Un mensaje en transmisión puede consistir en los atributos "De (dirección), A (dirección), Contenidos".*

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 todo lo que se requiere, desde la perspectiva de los Requisitos Funcionales del Usuario, para describir el objeto de interés. En la práctica, estos casos ocurren comúnmente en software de aplicación en tiempo real (por ejemplo, el grupo de datos ingresado para transmitir el tic de un reloj de tiempo real o la entrada del estado de un sensor); Son menos comunes en el software de aplicación empresarial.

3.5 Identificación de 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 movimiento de datos

DEFINICIÓN – Movimiento de Datos.
Un componente funcional básico que mueve un solo tipo de grupo de datos. NOTA 1: Hay cuatro subtipos de un movimiento de datos (tipo), a saber: Entrada, Salida, Lectura y Escritura (tipos). NOTA 2: Para fines de medición, se considera que cada movimiento de datos tiene en cuenta ciertas manipulaciones de datos asociadas; consulte la sección 3.5.6 para obtener 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 también se aplica a las definiciones de Entrada, Salida, Lectura y Escritura.

DEFINICIÓN – Entrada (E)
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 requiere. NOTA: Se considera que una entrada tiene en cuenta ciertas manipulaciones de datos asociadas (consulte la sección 3.5.6).

DEFINICIÓN – Salida (X)
Un movimiento de datos que mueve un grupo de datos de un proceso funcional a través de la frontera al usuario funcional que lo requiere. NOTA: Se considera que una salida tiene en cuenta ciertas manipulaciones de datos asociadas (consulte la sección 3.5.6).

DEFINICIÓN – Lectura (R)
Un movimiento de datos que mueve un grupo de datos del almacenamiento persistente al proceso funcional que lo requiere. NOTA: Se considera que una lectura es responsable de cierta manipulación de datos asociada (consulte la sección 3.5.6).

DEFINICIÓN – Escritura (W)
Un movimiento de datos que mueve un grupo de datos desde un proceso funcional a un almacenamiento persistente. NOTA Se considera que una escritura tiene en cuenta cierta manipulación de datos asociada (consulte la sección 3.5.6).

La Figura 3.5, a continuación, ilustra la relación general entre los cuatro tipos de movimiento de datos, el proceso funcional al que pertenecen y la frontera del software medido.

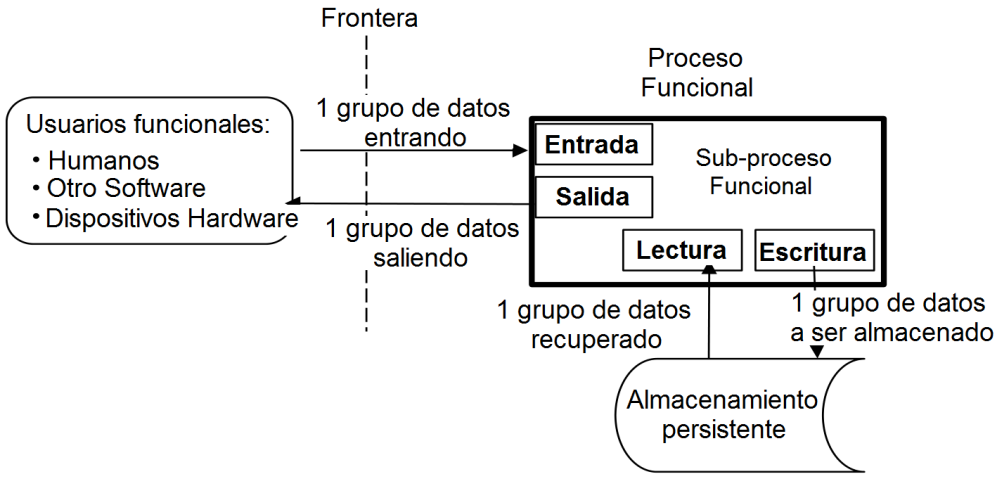


Figura 3.5 - Los cuatro tipos de movimiento de datos, cada uno moviendo un grupo de datos, y su relación con un proceso funcional. (Un proceso funcional puede, por supuesto, tener muchos movimientos de datos E, X, R y W.)

3.5.2 Entradas de identificación (E)

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

PRINCIPIOS – Entrada (E)
a) Una Entrada moverá un solo grupo de datos que describa un solo objeto de interés de un usuario funcional a través del límite y al proceso funcional del cual forma parte la Entrada. Si la entrada a un proceso funcional comprende más de un grupo de datos, cada uno describe un objeto de interés diferente, identifique una Entrada para cada grupo de datos único en la entrada. (Consulte también la sección 3.5.7 sobre "Movimientos de datos únicos".)
b) Una entrada no debe salir de los datos a través del límite, o leer o escribir datos desde/hacia el almacenamiento persistente.

Las siguientes reglas ayudan a confirmar el estado de un movimiento de datos de Entrada candidato:

REGLAS – Entrada (E)
a) El grupo de datos de una Entrada desencadenante puede consistir en un solo atributo de datos que simplemente informa al software que "ha ocurrido un evento Y". Muy a menudo, especialmente en el software de aplicación empresarial, el grupo de datos de la Entrada desencadenante tiene varios atributos de datos que informan al software que "ha ocurrido un evento 'Y' y aquí están los datos sobre ese evento en particular".
b) Las señales de reloj que desencadenan eventos siempre serán externas al software que se está midiendo. Por lo tanto, por ejemplo, un evento de marca de reloj que ocurre cada 3 segundos se asociará con una Entrada que mueve un grupo de datos de un atributo de datos. Tenga en cuenta que no hace ninguna diferencia si el evento desencadenante se genera periódicamente por hardware o por otra pieza de software fuera del límite del software que se mide.
c) A menos que sea necesario un proceso funcional específico, no se considerará que la obtención de la fecha y/o la hora del reloj del sistema causan una Entrada o cualquier otro movimiento de datos.

d) Si una ocurrencia de un evento específico causa la Entrada de un grupo de datos que comprende hasta 'n' atributos de datos de un objeto de interés en particular y el FUR permite que otras ocurrencias del mismo evento puedan causar una Entrada de un grupo de datos que tenga valores para los atributos de solo un subconjunto de los atributos 'n' del objeto de interés, se identificará una Entrada, moviendo un grupo de datos que comprende todos los atributos de datos 'n'.

e) Al identificar Entradas en una pantalla que permite a los usuarios humanos funcionales ingresar datos en procesos funcionales, analice solo las pantallas que están llenas de datos. Ignore cualquier pantalla que esté formateada, pero, por lo demás, "en blanco", excepto los posibles valores predeterminados, e ignore todos los encabezados de campo y otros que permiten a los usuarios humanos comprender los datos de entrada requeridos.

NOTA. Puede ser necesario tener en cuenta los encabezados de campo y otros al medir FUR para los cambios en las Entradas; consulte la sección 4.4.1.

EJEMPLO DE NEGOCIO que ilustra la regla c): cuando un proceso funcional agrega una marca de tiempo a un registro para que sea persistente o se genere, no se identifica ninguna Entrada para obtener el valor de reloj del sistema.

Una vez identificada, cada movimiento de datos de Entrada puede registrarse marcando la celda correspondiente de la matriz del Modelo Genérico de Software (Apéndice A) con una 'E'.

3.5.3 Identificando Salidas (X)

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

PRINCIPIOS – Salida (X)

a) Una Salida moverá un solo grupo de datos que describa un único objeto de interés del proceso funcional del cual la Salida forma parte a través de la frontera a un usuario funcional. Si la salida de un proceso funcional comprende más de un grupo de datos, identifique una Salida para cada grupo de datos único en la salida. (Consulte también la sección 3.5.7 sobre "Movimientos de datos únicos".)

b) Una Salida no debe ingresar datos a través de la frontera, o leer o escribir datos desde/hacia el almacenamiento persistente.

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

REGLAS - Salida (X)

a) Una consulta que genere texto fijo, (donde 'fijo' significa que el mensaje no contiene valores de datos variables, por ejemplo, el resultado de presionar un botón para 'Términos y condiciones' en un sitio web de compras), se modelará con una Salida para la salida de texto fija.

   NOTA: Para obtener información sobre la salida de la funcionalidad ‘Ayuda’, consulte la ‘Guía para dimensionar el Software de aplicación empresarial’. Para la salida de mensajes relacionados con condiciones de error o confirmación de éxito, consulte la sección 3.5.11 de este Manual de medición.

b) Si una Salida de un proceso funcional mueve un grupo de datos que comprende hasta 'n' atributos de datos de un objeto de interés en particular y el FUR permite que el proceso funcional tenga una ocurrencia de una Salida que mueva un grupo de datos que tenga valores para atributos de solo un subconjunto de los atributos 'n' del objeto de interés, se identificará una Salida, moviendo un grupo de datos que comprende todos los atributos de datos 'n'.

c) Al identificar Salidas, ignore todos los campos y otros encabezados que permitan a los usuarios humanos comprender los datos de salida.

NOTA: Es posible que sea necesario tener en cuenta los encabezados de campo y otros al medir FUR para cambios en Salidas; consulte la sección 4.4.1

Una vez identificada, cada movimiento de datos de salida puede registrarse marcando la celda correspondiente de la matriz del Modelo Genérico de Software (Apéndice A) con una "X".

Consulte también la sección 3.5.11 para saber cómo identificar los movimientos de datos de Salida para mensajes de error.

3.5.4 Identificando Lectura (R)

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

PRINCIPIOS - Lectura (R)
a) Una Lectura moverá un solo grupo de datos que describa un único objeto de interés desde el almacenamiento persistente a un proceso funcional del que forma parte la Lectura. Si el proceso funcional debe recuperar más de un grupo de datos del almacenamiento persistente, identifique una Lectura para cada grupo de datos único que se recupere. (Consulte también la sección 3.5.7 sobre "Movimientos de datos únicos".)
b) Una Lectura no recibirá ni enviará datos a través de su frontera, ni escribirá datos en el almacenamiento persistente.
c) Durante un proceso funcional, el movimiento o la manipulación de constantes o variables que son internas al proceso funcional y que pueden ser cambiadas solo por un programador, o el cálculo de resultados intermedios en un cálculo, o de datos almacenados por un proceso funcional que resulta solo de la implementación, en lugar de la FUR, no se considerará como movimientos datos de Lectura.
d) Un movimiento de datos de Lectura siempre incluye cualquier función de "solicitud de Lectura" (por lo tanto, un movimiento de datos por separado nunca se contabilizará para cualquier funcionalidad de "solicitud de Lectura"). Ver también la sección 3.5.9.

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

REGLAS - Lectura (R)
a) Identifique una Lectura cuando, según el FUR, el software que se está midiendo debe recuperar un grupo de datos del almacenamiento persistente.
b) No identifique una Lectura cuando el FUR del software que se está midiendo especifique a cualquier usuario funcional de software o hardware como la fuente de un grupo de datos, o como el medio de recuperar un grupo de datos almacenado de forma persistente. (Para este caso, vea los principios y reglas para Entradas y Salidas).

Una vez identificado, cada movimiento de datos de Lectura se puede registrar marcando la celda correspondiente de la matriz del Modelo Genérico de Software (Apéndice A) con una "R".

3.5.5 Identificando Escrituras (W)

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

PRINCIPIOS – Escritura (W)

a) Una Escritura moverá un solo grupo de datos que describa un único objeto de interés del proceso funcional del cual la Escritura forma parte del almacenamiento persistente. Si el proceso funcional debe mover más de un grupo de datos al almacenamiento persistente, identifique una Escritura para cada grupo de datos único que se mueva al almacenamiento persistente. (Consulte también la sección 3.5.7 sobre "Movimientos de datos únicos".)

b) Una Escritura no recibirá o enviará datos a través de la frontera, ni leerá datos del almacenamiento persistente.

c) El requisito de eliminar un grupo de datos del almacenamiento persistente se medirá como un solo movimiento de datos de Escritura.

d) Lo siguiente no se considerará como un movimiento de datos de Escritura:

- El movimiento o manipulación de cualquier dato que no existía al inicio de un proceso funcional y que no se haya hecho persistente cuando el proceso funcional está completo;
- Creación o actualización de variables o resultados intermedios que son internos al proceso funcional;
- Almacenamiento de datos por un proceso funcional que resulta solo de la implementación, en lugar del FUR. (Un ejemplo sería el almacenamiento de datos temporalmente durante un proceso de clasificación grande en un trabajo procesado por lotes).

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

REGLAS – Escritura (W)

a) Identifique una Escritura cuando, de acuerdo con el FUR, el software que se está midiendo debe mover un grupo de datos a un almacenamiento persistente.

b) No identifique una Escritura cuando el FUR del software que se está midiendo especifique a cualquier usuario funcional de software o hardware como destino del grupo de datos o como medio para almacenar el grupo de datos. (Para este caso, vea los principios y reglas para Entradas y Salidas).

Una vez identificado, cada movimiento de datos de Escritura se puede registrar marcando la celda correspondiente de la matriz del Modelo de software genérico (Apéndice A) con una "W".

3.5.6 Sobre las manipulaciones de datos asociadas a movimientos de datos.

Los subprocesos son, como se define en el principio (d) del Modelo Genérico de Software (consulte la sección 1.3), ya sea movimientos de datos o manipulaciones de datos. Sin embargo, por una convención COSMIC (ver principio (j) del Modelo Genérico de Software), no se reconoce la existencia separada de subprocesos de manipulación de datos.

Se considera que toda la manipulación de datos se debe a los movimientos de datos asociados. Por lo tanto, la manipulación de datos puede ignorarse EXCEPTO si hay un FUR que debe medirse para un *cambio* en la manipulación de datos. Para estos casos, es posible que necesitemos las reglas de esta sección para determinar qué tipo de manipulación de datos está asociada con qué tipo de movimiento de datos.

Una solicitud de cambio típica afecta tanto a los atributos de datos movidos como a la manipulación de datos asociada con un movimiento de datos en particular. Las reglas para medir los cambios requeridos del FUR (consulte la sección 4.4 y la definición de 'Modificación' en 4.4.1) son que, si algún atributo del grupo movido por un movimiento de datos o cualquiera de sus manipulaciones de datos asociados debe ser cambiado, entonces el movimiento de datos debe ser identificado y medido como cambio. Entonces, si un cambio requerido afecta solo

a la manipulación de datos, necesitamos las siguientes reglas para identificar sus movimientos de datos asociados que deben medirse como cambiados,

DEFINICIÓN – Manipulación de Datos
Cualquier cosa que suceda con los datos cuando se procesa mediante un proceso funcional distinto del movimiento de datos dentro o fuera de un proceso funcional, o entre un proceso funcional y el almacenamiento persistente.

El siguiente principio y las reglas determinan cómo el método COSMIC trata la manipulación de datos.

PRINCIPIO - Manipulación de datos asociada a movimientos de datos.
Toda la manipulación de datos en un proceso funcional se asociará con los cuatro tipos de movimiento de datos (E, X, R y W). Por convención, se supone que los movimientos de datos de un proceso funcional también tienen en cuenta la manipulación de los datos del proceso funcional.

REGLAS - Manipulación de datos asociada a movimientos de datos.
a) Un movimiento de datos de Entrada representa toda la manipulación de datos para permitir que un usuario funcional ingrese un grupo de datos (por ejemplo, manipulaciones de formato y presentación) y que se validen.
b) Un movimiento de datos Salida representa toda la manipulación de datos para crear los atributos de datos de un grupo de datos que se deben generar y/o permitir que el grupo de datos se genere (por ejemplo, manipulaciones de presentación y formato) y que se dirijan al usuario funcional deseado.
c) Un movimiento de datos de Lectura representa todos los cálculos y/o procesamientos lógicos necesarios para recuperar un grupo de datos del almacenamiento persistente.
d) Un movimiento de datos de Escritura representa todos los cálculos y/o procesamientos lógicos para crear o actualizar un grupo de datos a ser escrito, o para eliminar un grupo de datos.
e) La manipulación de datos asociada con cualquiera de estos movimientos de datos no incluye ninguna manipulación de datos que sea necesaria después de que el movimiento de datos se haya completado con éxito, ni incluye ninguna manipulación de datos asociada con ningún otro movimiento de datos.

EJEMPLO DE NEGOCIO 1: Una Entrada incluye toda la manipulación necesaria para formatear una pantalla para permitir que un usuario humano ingrese datos y valide los datos ingresados EXCEPTO cualquier Lectura (s) que pueda ser necesaria para validar algunos datos o códigos ingresados, u obtener algunas descripciones de códigos asociados.

EJEMPLO DE NEGOCIO 2: Una Salida incluye toda la manipulación para dar formato a la salida y preparar algunos atributos de datos para imprimir (o salida en una pantalla), incluidos los encabezados[11] de los campos legibles por humanos, EXCEPTO cualquier Lectura o Entrada que pueda ser necesaria para proporcionar los valores o descripciones de algunos de los atributos de datos impresos.

3.5.7 Movimientos de datos únicos y posibles excepciones.

En la mayoría de los casos, en cualquier proceso funcional, *todos* los datos que describen un objeto de interés que requiere ese proceso funcional, se ingresan en un tipo de movimiento de datos de Entrada y/o se leen en un

[11] Este ejemplo se aplica cuando se mide software de aplicación para uso humano, independientemente del dominio. Obviamente, no se aplicaría al medir el tamaño de los objetos reutilizables que admiten la visualización de encabezados de campo individuales en las pantallas de entrada o salida.

tipo de movimiento de datos de Lectura y/o se escriben en un tipo de movimiento de datos de Escritura y/o salen en un tipo de movimiento de datos de 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 mueven está asociada con un movimiento de datos. Además, dos movimientos de datos no pueden considerarse diferentes debido a que tienen FUR diferente para su manipulación de datos asociada, ya que 'toda manipulación de datos en un proceso funcional se asociará con los cuatro tipos de movimiento de datos (E, X, R y W) '(Ver el principio establecido en la sección 3.5.6).

EJEMPLO que ilustra este último principio: Considere dos ocurrencias en un proceso funcional dado (tipo). Supongamos que en la primera aparición los valores de algunos atributos que se moverán conducen a un subproceso de manipulación de datos (-tipo) 'A' y que, en otra aparición del mismo proceso funcional, los valores de los atributos conducen a un sub- proceso (-tipo) 'B'. En tales circunstancias, tanto los subprocesos de manipulación de datos "A" como "B" deben asociarse con el mismo movimiento de datos y, por lo tanto, solo el movimiento de datos debe identificarse y contabilizarse en ese proceso funcional.

Sin embargo, puede haber circunstancias en las que se requiera (en el FUR) que *diferentes* grupos de datos que describen el *mismo* objeto de interés se muevan en un movimiento de datos del mismo tipo (E, R, W, X) en el mismo proceso funcional.

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

REGLAS - Movimientos de datos únicos y posibles excepciones.

Nótese bien que todas las reglas COSMIC 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. Esta convención se sigue en las reglas a), b) y c) a continuación, pero en la regla d) incluimos "tipo" donde es útil distinguir un "tipo" de una "ocurrencia".

a) A menos que los Requisitos de Usuario Funcional sean los que se indican en las reglas b) o c), todos los datos que describan cualquier objeto de interés que deba ingresarse en un proceso funcional se identificarán como un grupo de datos movido por una Entrada.

   NOTA: un proceso funcional puede, por supuesto, tener múltiples Entradas, cada una de ellas con datos en movimiento que describen un objeto de interés diferente.

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

b) Si los Requisitos Funcionales del Usuario especifican que se deben ingresar diferentes grupos de datos en un proceso funcional, uno por cada usuario funcional diferente, donde cada grupo de datos describe el mismo objeto de interés, se identificará una Entrada para cada uno de estos grupos de datos diferentes.

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

   NOTA: Cualquier proceso funcional tendrá una sola Entrada desencadenante.

c) Si los Requisitos Funcionales del Usuario especifican que se deben mover diferentes grupos de datos del almacenamiento persistente a un proceso funcional, cada uno describiendo el mismo objeto de interés, se identificará una Lectura para cada uno de estos grupos de datos diferentes.

   La misma regla equivalente se aplica para Escrituras en cualquier proceso funcional dado.

   NOTA: Esta regla es análoga a la regla b). En el caso de los FUR para leer diferentes grupos de datos que describen el mismo objeto de interés, probablemente se hayan originado a partir de diferentes usuarios funcionales. En el caso de que los FUR escriban diferentes grupos de datos, es probable que estén disponibles para que los lean diferentes usuarios funcionales.

d) Las ocurrencias repetidas de cualquier tipo de movimiento de datos cuando se está ejecutando no se contabilizarán.

Esto se aplica incluso si varias ocurrencias del tipo de movimiento de datos difieren en su ejecución debido a que diferentes valores de los atributos de datos del grupo de datos movidos dan como resultado diferentes caminos de procesamiento que se siguen a través del tipo de proceso funcional.

Los siguientes ejemplos ilustran las reglas anteriores.

EJEMPLO DE NEGOCIO 1 para la regla a): El caso más común es que se identifique una Escritura que mueva un grupo de datos que contenga <u>todos</u> los atributos de datos de un objeto de interés requerido que deben hacerse persistentes en un proceso funcional dado.

EJEMPLO DE NEGOCIO 2 para la regla c) en contraste con la regla a) y el Ejemplo de negocio 1: Suponga un FUR para que un proceso funcional único A almacene dos grupos de datos derivados de los archivos de cuentas actuales de un banco para su uso posterior por programas separados. El primer grupo de datos es "detalles de la cuenta sobregirada" (que incluye el atributo de saldo negativo). El segundo grupo de datos es "detalles de la cuenta de alto valor" (que solo tiene el nombre y la dirección del titular de la cuenta, destinados a una toma de correo de marketing). El proceso funcional A tendrá dos Escrituras, una para cada grupo de datos, ambas que describen el mismo objeto de interés "cuenta".

EJEMPLO EN TIEMPO REAL 3 para la regla b): Se requiere un proceso funcional para aceptar diferentes grupos de datos de dos sismómetros diferentes (usuarios funcionales), cada uno de los cuales describe el mismo objeto de interés (el evento), por ejemplo. Una explosión de prueba. Identificar dos entradas.

EJEMPLO DE NEGOCIO 4 para la regla b): Suponga que existe un FUR para que un proceso funcional único produzca dos o más Salidas que mueven diferentes grupos de datos que describen el mismo objeto de interés, destinado a diferentes usuarios funcionales. Por ejemplo, cuando un empleado nuevo se une a una empresa, se produce un informe que se pasa al empleado para firmar sus datos personales como válidos, y se envía un mensaje a Seguridad para autorizar al empleado a ingresar al edificio. Identificar dos Salidas.

EJEMPLO DE NEGOCIO 5 para la regla c): Suponga que existe un FUR para un programa que fusiona dos archivos de datos persistentes que describen el mismo objeto de interés, por ejemplo, un archivo de datos existentes sobre un objeto de interés "X" y un archivo con atributos recién definidos que describen el mismo objeto de interés "X". Identifique dos Lecturas, una para cada archivo, para el proceso funcional.

EJEMPLO DE NEGOCIO 6 para la regla c): Supongamos que se requiere una Lectura de un grupo de datos en el FUR, pero el desarrollador decide implementarlo mediante dos comandos para recuperar diferentes subconjuntos de atributos de datos del mismo objeto de interés del almacenamiento persistente en Diferentes puntos en el proceso funcional. Identifique solo una Lectura.

EJEMPLO DE NEGOCIO 7 para la regla d): Consulte la sección 3.5.2, regla d) para las Entradas, y la sección 3.5.3, regla b) para las Salidas. (Estas dos reglas se refieren a los casos en que las ocurrencias de Entradas o Salidas tienen solo un subconjunto del número máximo de tipos de atributos de datos que se pueden mover de acuerdo con el FUR).

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

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

EJEMPLO EN TIEMPO REAL 10 para la regla d): Supongamos un sistema de control de proceso para una máquina que produce un producto plano como papel o una película plástica. La máquina tiene una serie de

100 sensores idénticos en dirección del movimiento del producto para detectar roturas u orificios en el producto. El proceso funcional que debe verificar si hay roturas o agujeros recibe los mismos datos de cada sensor. La posición de, por ejemplo, un agujero en el producto se puede determinar, ejemplo, desde la posición en la cadena de valores de datos enviados desde la matriz de sensores, o por cada sensor que envía su ID, etc. De cualquier manera, el procesamiento de datos de todos los sensores es idéntico. Identifique solo un tipo de usuario funcional para todos los sensores y solo un tipo de Entrada para los datos obtenidos de todos los sensores por el proceso funcional.

3.5.8 Cuando se requiere un proceso funcional para mover datos hacia o desde el almacenamiento persistente.

Cuando el FUR requiere que los datos se almacenen o se recuperen del almacenamiento, el Medidor debe investigar si los datos pueden almacenarse o recuperarse dentro de su propio límite, es decir, hacia/desde el 'almacenamiento persistente', o si se requiere que los datos se almacenen/recuperen con la ayuda de un usuario funcional del software que se está midiendo (es decir, a través de algún otro software, o directamente a o desde un dispositivo de hardware). Esta sección describe los movimientos de datos involucrados cuando se requiere un proceso funcional de una pieza de software para obtener algunos datos del almacenamiento persistente, a través de cuatro ejemplos a continuación:

- Ejemplo 1 es típico del software de aplicación, pero podría aplicarse al software en cualquier capa, excepto en la capa donde el software interactúa directamente con una tienda física de hardware. El proceso funcional es necesario para recuperar datos de un almacenamiento persistente, pero el FUR no está preocupado por la forma en que esos accesos a los datos son manejados por cualquier otro software;
- Ejemplo 2 es para el software con una arquitectura de "cliente-servidor" donde un proceso funcional de un cliente debe solicitar al servidor 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 se deben obtener directamente de una tienda física de hardware, tal vez mediante el software del controlador del dispositivo.

Los ejemplos se ilustran utilizando las convenciones de los diagramas de secuencia de mensajes. La notación de estos diagramas es la siguiente:

- Una flecha vertical en negrita que apunta hacia abajo representa un proceso funcional.
- Las flechas horizontales representan movimientos de datos, etiquetados E, X, R o W para Entrada, Salida, Lectura y Escritura, respectivamente. Las Entradas y Lecturas se muestran como flechas entrantes al proceso funcional y las Salidas y Escrituras como flechas salientes; en la medida de lo posible, aparecen en la secuencia requerida, de arriba a abajo, del proceso funcional.
- Una línea punteada vertical representa la frontera

  EJEMPLO 1: Cuando se requiere un proceso funcional para mover un grupo de datos hacia o desde el almacenamiento persistente. Este ejemplo se refiere a una parte del software A que se requiere para recuperar un grupo de datos almacenados en los que el FUR del software "A" no está preocupado por la forma en que esos accesos a los datos son manejados por cualquier otro software en la misma o en diferentes capas.

  Los usuarios funcionales del software A podrían ser, por ejemplo, usuarios humanos si el software A está en la capa de aplicación realizando una consulta sobre un grupo de datos almacenados. La Figura 3.6 muestra los movimientos de datos de COSMIC para esta consulta. La consulta se inicia mediante una Entrada, seguida de una Lectura del grupo de datos del almacenamiento persistente y luego una Salida con el resultado de la consulta. FP A no se ocupa de dónde se recuperan los datos, solo que son datos persistentes.

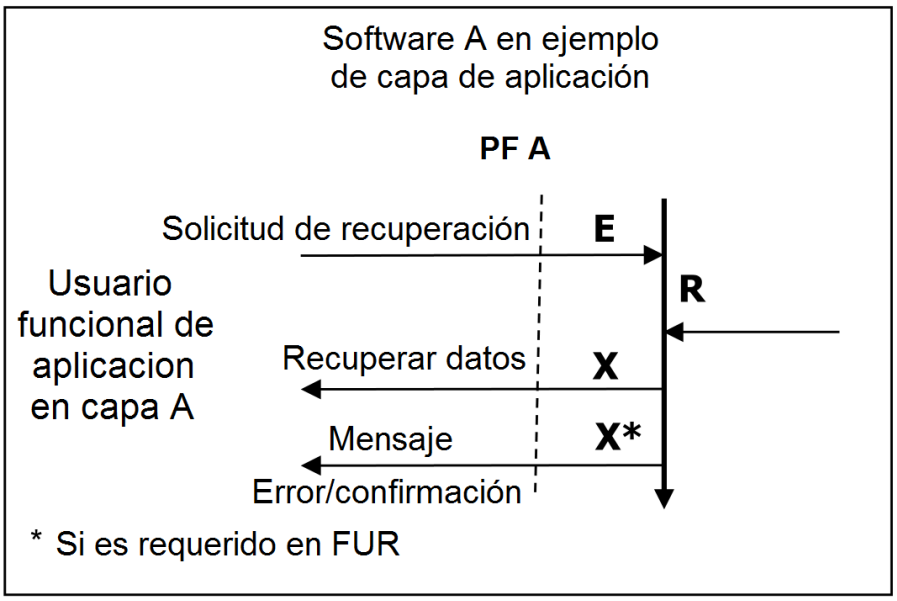


Figura 3.6 - Solución para una lectura emitida por el software 'A' en la capa de aplicación

Se aplicaría un modelo exactamente análogo si se requiriera que el proceso funcional PF A hiciera un grupo de datos persistente a través de un movimiento de datos de Escritura. De acuerdo con la regla d) para condiciones de error (consulte la sección 3.5.11), se considera que los movimientos de datos de Lectura y Escritura tienen en cuenta cualquier código de retorno o informe de una condición de error.

La Figura 3.6 muestra un posible mensaje de error específico de la aplicación que podría emitir el PF A si, por ejemplo, no se encuentra el registro solicitado. Sin embargo, una condición de error no es específica de la aplicación, por ejemplo, "Falla de disco" no se contaría como una Salida para el PF A. Consulte también la sección 3.5.11 sobre mensajes de error/confirmación.

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

En este ejemplo, se asume que las piezas de software que se medirán tienen una relación 'cliente/servidor', es decir, donde una pieza, el cliente, obtiene 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 de este tipo, en la que las dos piezas son componentes principales de la misma aplicación. En cualquier relación cliente/servidor, el FUR del componente C1 del cliente identificaría al componente C2 del servidor como uno de sus usuarios funcionales, y viceversa. La misma relación existiría y el mismo diagrama se aplicaría si las dos piezas fueran aplicaciones separadas, 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, intercambiarían datos a través de los sistemas operativos respectivos y cualquier otra capa intermedia de sus procesadores en una arquitectura de software como la que 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 hardware que intervienen se ignoran en este modelo. (Vea también el lado derecho de la Figura 3.1 para un ejemplo similar).

Teniendo en cuenta el posible mensaje de error/confirmación emitido por el cliente, esta consulta del Ejemplo 2, por lo tanto, requerida de 5 movimientos de datos (es decir, 5 CFP) para satisfacer la solicitud de consulta para el componente C1 y 3 CFP para el componente C2. Esto se compara con los 4 CFP (1 x E, 1 x R y 2 x X) que se habrían requerido para el componente C1 si hubiera podido recuperar el grupo de datos del almacenamiento persistente dentro de su propio límite a través de una Lectura como se muestra en Figura 3.7.

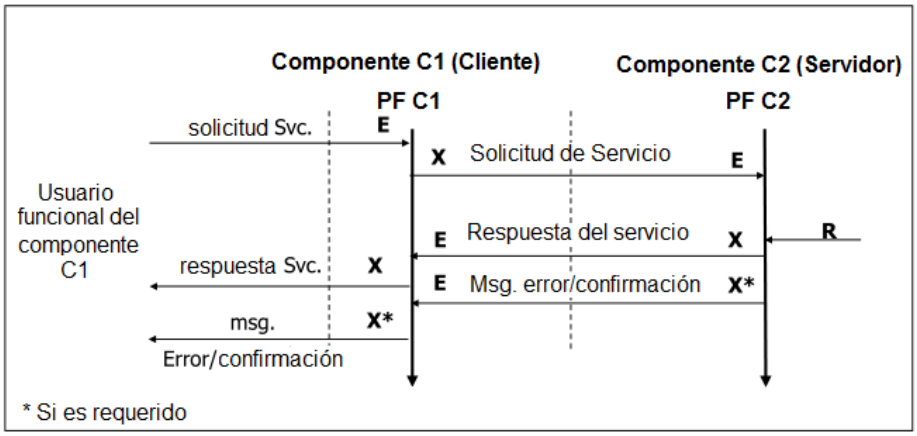


Figura 3.7 - Intercambios de datos entre componentes de cliente y servidor

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

Los ejemplos 1 y 2 ilustran los movimientos de datos cuando el FUR aclara que el software que se está midiendo debe acceder al almacenamiento persistente dentro de su propio límite, o debe pasar la solicitud de acceso a otro software fuera de su límite, respectivamente. A veces, sin embargo, la pieza de software que se está midiendo puede tener que usar diferentes "rutas" para acceder a los datos persistentes dependiendo de los atributos de datos específicos a los que se debe acceder y / o el tipo de acceso (almacenamiento o recuperación). Esto puede surgir cuando el acceso a los datos por parte del software que se está midiendo está sujeto a diferentes reglas o "derechos" debido a problemas, por ejemplo, seguridad o privacidad (consulte el Ejemplo 3 a continuación), o la necesidad de garantizar la integridad de los datos restringiendo el acceso a todos los procesos de creación, actualización y eliminación. Cuando el FUR del software que se está midiendo no está claro en este punto, el Medidor debe tener cuidado para determinar los derechos reales de acceso. (NO confunda "derecho de acceso" a los datos con "propiedad" de datos; este último es irrelevante para el modelo COSMIC. El almacenamiento persistente no es "propiedad" de ninguna pieza de software).

Nota: el ejemplo de la Figura 3.7 supone una comunicación sincrónica entre el cliente y el servidor, es decir, el cliente espera la respuesta del servidor. Alternativamente, la comunicación podría tener lugar de forma asincrónica. Este caso se modelaría de manera ligeramente diferente, pero el número de movimientos de datos sería el mismo que en la Figura 3.7 (consulte la "Guía para dimensionar el software de aplicación empresarial" [7]).

Nota: en la práctica, la comunicación cliente-servidor puede estar sujeta a monitoreo de tiempo de espera en caso de que el servidor no responda dentro de un tiempo requerido. Para ver un ejemplo de cómo se puede modelar y medir esto, consulte la "Guía para dimensionar el software en tiempo real" [4].

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

Ver Figura 3.8. Se permite que una pieza de software A para medir recupere ciertos datos Z almacenados (como en el Ejemplo 1, Figura 3.6), pero no se permite mantener (es decir, crear, actualizar o eliminar) estos mismos datos Z directamente. Cuando se requiere que el software A mantenga los datos Z, el software A debe pasar su solicitud al software B a través de una Salida seguida de una Entrada (análoga al Ejemplo 2, Figura 3.7 del componente C1 que pasa su solicitud al componente C2). El software B es necesario para garantizar la integridad de los datos Z garantizando una validación coherente, por lo que procesa todas las solicitudes de mantenimiento de datos para los datos Z.

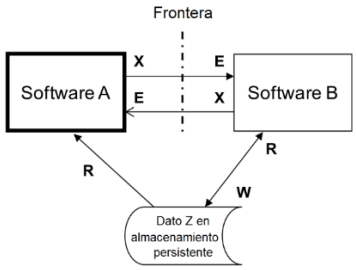


Figura 3.8 - Datos persistentes Z dentro de los límites del software A y B para una lectura

En este ejemplo 3, los modelos COSMIC mostrarían que los datos Z se almacenan en un almacenamiento persistente dentro del límite del software A, pero solo para fines de recuperación a los que se puede acceder mediante movimientos de datos de Lectura. Para el software B, estos mismos datos Z se mantienen en el almacenamiento persistente dentro de su límite y el software B puede Leer y Escribir estos datos Z. Para el manejo de errores en este ejemplo, consulte los ejemplos 1 y 2 anteriores.

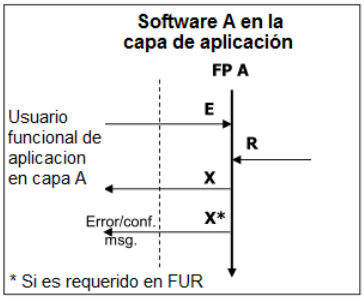
EJEMPLO DE INFRAESTRUCTURA 4: Cómo el software del controlador del dispositivo obtiene datos persistentes que interactúan con el dispositivo de almacenamiento físico.

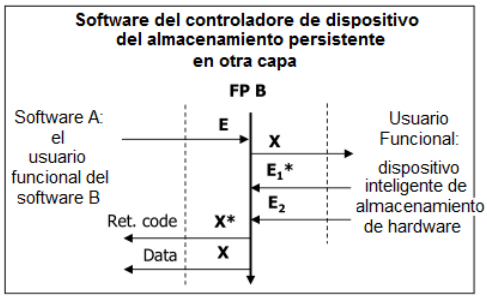
Este ejemplo se refiere a la pieza de software A del Ejemplo 1 que se requiere para recuperar un grupo de datos almacenado. También consideramos una pieza separada de software "B" que es el controlador del dispositivo para el almacenamiento inteligente de hardware que contiene el grupo de datos al que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

Las dos piezas de software están en diferentes capas en una arquitectura como se muestra en la Figura 2.2. El software A está, p. ej. en la capa de aplicación y el software B está en una capa de controlador de dispositivo. Físicamente, es probable que haya una relación jerárquica entre las dos piezas y (ignorando el sistema operativo) 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 del software A y B son independientes de la naturaleza de la relación entre las capas, que pueden ser jerárquicas o bidireccionales.

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

Suponga que un proceso funcional de consulta PF A del software A necesita recuperar un grupo de datos almacenado. La Figura 3.9 (a) muestra el modelo COSMIC de esta investigación. La Figura 3.9 (b) muestra el proceso funcional FP B del software B en la capa del controlador del dispositivo que maneja la recuperación física de los datos requeridos desde un dispositivo de almacenamiento de hardware (como un disco o una memoria USB).





Figuras 3.9 (a) y (b) - Solución para una lectura emitida por el software A en la capa de aplicación al software B en la capa del controlador del dispositivo.

La figura 3.9 (b) muestra que la solicitud de Lectura del software A se recibe como una Entrada Desencadenante del proceso funcional PF B, que pasa la solicitud como una Salida al dispositivo de hardware. La respuesta de este último depende del dispositivo de hardware particular. El dispositivo puede simplemente devolver los datos solicitados, que se muestran como Entrada E_2 en la 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, p. ej. "datos no encontrados" o "error de disco", que se muestran como Entrada E_1 en la Figura 3.9 b). El PF B devuelve los datos al software A como una Salida. El PF B normalmente también emite un "código de retorno" que describe el éxito o la razón del fracaso de la solicitud. (Aunque el código de retorno puede estar físicamente adjunto a los datos devueltos, lógicamente es un grupo de datos diferente al de los datos devueltos; son datos sobre el resultado del proceso de solicitud). Para el PF A, no se identifica ninguna Entrada para estos mensajes, ya que el movimiento de datos de Lectura representa los datos recuperados y los mensajes de error, de acuerdo con la regla d) para una Entrada. Para el PF A, se identifica una Salida para un mensaje de error/confirmación, 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 los que se muestran en la figura 3.8 b). Por ejemplo, esta Figura no muestra el efecto del controlador del dispositivo que mide un tiempo de espera por 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 se debe a que A y B están en capas diferentes y las Lecturas no cruza un límite. La Figura 3.9 (b) muestra que el software A es un usuario funcional del software del controlador del dispositivo B. Pero lo contrario no es cierto, porque una Lectura no cruza un límite. En contraste, la Figura 3.7 puede mostrar los dos componentes en un modelo porque el Componente C1 es un usuario funcional del componente C2, y viceversa, y comparten un límite 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 necesita decirle al usuario funcional qué datos enviar, una sola Entrada es suficiente (por objeto de interés). Si el proceso funcional necesita decirle al usuario funcional qué datos enviar, es necesaria una Salida seguida de una Entrada. Se aplican las siguientes reglas:

REGLAS - Un proceso funcional que requiere datos de un usuario funcional

a) Un proceso funcional debe obtener un grupo de datos a través de un movimiento de datos de Entrada de un usuario funcional, cuando el proceso funcional no necesita decirle al usuario funcional qué datos 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 el proceso funcional;
- cuando un proceso funcional, después de haber recibido un grupo de datos a través de una Entrada desencadenante, espera, expectante la llegada de un grupo de datos adicional del usuario funcional a través de una Entrada (puede ocurrir cuando un usuario funcional humano ingresa datos al software de aplicación de negocios);
- cuando un proceso funcional, después de haber comenzado, solicita al usuario funcional, "envíeme sus datos ahora, si tiene alguno" y el usuario funcional envía sus datos;
- cuando un proceso funcional, después de haber comenzado, inspecciona el estado de un usuario funcional y recupera los datos que requiere.

En los últimos dos casos (que generalmente ocurren en el software de "sondeo" en tiempo real), por convención, no se identificará ninguna Salida del proceso funcional para obtener los datos requeridos. El proceso funcional simplemente necesita enviar un mensaje de solicitud a un usuario funcional y la funcionalidad de ese mensaje de solicitud se considera parte de la Entrada. El proceso funcional sabe qué datos esperar. Solo se identificará una Entrada para este caso.

b) Cuando un proceso funcional necesita obtener los servicios de un usuario funcional (por ejemplo, para obtener datos) y *el usuario funcional necesita que se le diga qué enviar* (generalmente donde el usuario funcional es otra pieza de software fuera del alcance del software que se está midiendo), se identificará una Salida seguida de un movimiento de datos de Entrada. La Salida envía la solicitud de los datos específicos; la Entrada recibe los datos devueltos.

EJEMPLO EN TIEMPO REAL 1 de la regla a), tercera o cuarta viñeta: Suponga que se requiere un proceso funcional de un sistema de software de aplicación de control de proceso en tiempo real para sondear una matriz de sensores tontos idénticos. A nivel de aplicación, la solicitud de los datos por el proceso funcional y la recepción de los datos se contabiliza por una Entrada (tipo). (Dado que los sensores son idénticos, solo se identifica y cuenta una Entrada (tipo), aunque hay múltiples ocurrencias).

Supongamos además que la solicitud de datos debe pasarse en la práctica a una pieza de software de controlador de dispositivo en una capa inferior de la arquitectura de software, que físicamente obtiene los datos requeridos de la matriz de sensores como se ilustra en la arquitectura en capas de la Figura 2.3. Los procesos funcionales del software de aplicación de control de procesos y del software del controlador del dispositivo para los sensores tontos serían los que se muestran en las Figuras 3.10 (a) y (b) a continuación.

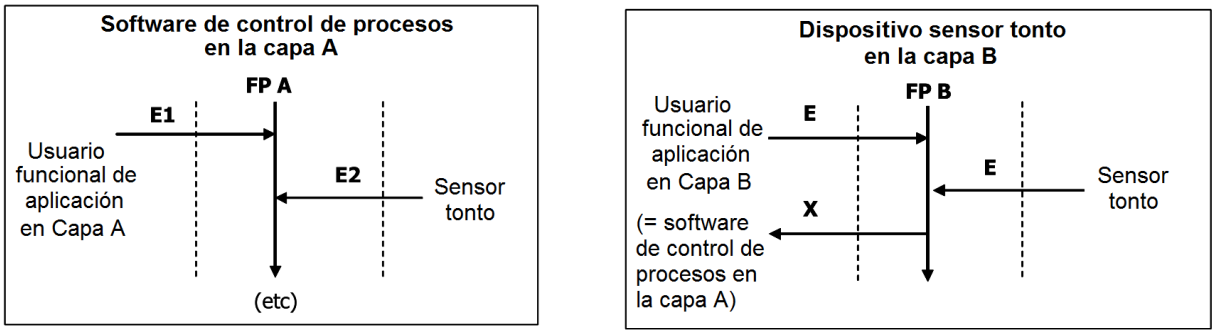


Figura 3.10 (a) y (b) - Solución para un sondeo de sensores tontos emitidos por el software A en la capa de aplicación de control de proceso manejada por el software B en la capa de controlador de dispositivo de sensor tonto.

La figura 3.10 (a) muestra que el proceso funcional PF A del software de aplicación es desencadenado mediante una Entrada E1, p. ej. de una marca de reloj. Este proceso funcional luego obtiene datos a través de la Entrada E2 del conjunto de sensores tontos para recibir las múltiples ocurrencias de las lecturas del sensor. Los sensores tontos también son usuarios funcionales del software de control de procesos en este modelo de nivel de aplicación. (El software del controlador del dispositivo está oculto en este nivel).

La Figura 3.10 (b) muestra el modelo para el software que maneja los dispositivos de sensores tontos. Recibe datos a través de una Entrada de la aplicación (probablemente en la práctica a través de un sistema operativo) como el detonador de un proceso funcional PF B. Este proceso funcional obtiene los datos requeridos a través de una Entrada E de su usuario funcional, el arreglo de sensores tontos.

El grupo de datos se devuelve al software de control de proceso a través de una Salida. Esta Salida se recibe como la Entrada E2 por el proceso funcional de la aplicación PF A. El PF A luego continúa con el procesamiento de los datos del sensor. Nuevamente, el hecho de que haya múltiples ocurrencias de este ciclo de recopilación de datos de cada uno de los sensores idénticos es irrelevante para el modelo.

La aparente falta de coincidencia entre la Entrada E2 de un sensor tonto al software de aplicación de control de proceso y la Entrada seguida de un movimiento de datos de Salida del software del controlador del dispositivo se debe a la convención de que se considera que una Entrada de un sensor tonto incluye cualquier funcionalidad de 'solicitud de ingreso' ya que el usuario funcional tonto no tiene la capacidad de manejar ningún mensaje de un proceso funcional.

EJEMPLO EN TIEMPO REAL 2 de la regla b): Suponga que un proceso funcional envía a uno de sus usuarios funcionales, como un dispositivo de hardware "inteligente" u otra pieza de software similar, algunos parámetros para una consulta o los parámetros para un cálculo, o algunos datos para ser comprimido. La respuesta del usuario funcional se obtiene a través del proceso funcional que emite una Salida, seguido de 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 control de navegación y visualización para usuarios humanos ("comandos de control")

Un "comando de control" es un comando que se reconoce en cualquier aplicación que pueda ser utilizado por usuarios funcionales humanos y que debe ignorarse al medir un tamaño funcional. La definición es:

DEFINICIÓN – Comando de control
Un comando que permite a los usuarios funcionales humanos controlar su uso del software pero que no implica ningún movimiento de datos sobre un objeto de interés del 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.

REGLA – Comandos de control en aplicaciones con una interfaz humana
En una aplicación con una interfaz humana, los "comandos de control" se ignorarán, ya que no implican ningún movimiento de datos sobre un objeto de interés.

EJEMPLOS DE COMANDOS DE CONTROL

- *Comandos para "subir / bajar página" o entre pantallas físicas.*
- *Al presionar una tecla Tab o Enter, o al presionar un botón para continuar.*
- *Al hacer clic en el botón "Aceptar/OK" para confirmar o cancelar una acción anterior, o para confirmar un mensaje de error o para confirmar algunos datos ingresados, etc.*
- *Funciones que permiten a un usuario controlar la visualización (o no) de un encabezado o de subtotales que se han calculado;*
- *Comandos de menú que permiten al usuario navegar a uno o más procesos funcionales específicos pero que no inician ningún proceso funcional,*
- *Comandos para mostrar una pantalla vacía para la entrada de datos.*

Nótese bien. Fuera del dominio de las aplicaciones con una interfaz de usuario humana, el concepto de un "comando de control" no tiene un significado especial y cualquier señal o movimiento de datos sobre un objeto de interés proveniente de un usuario funcional debe tenerse en cuenta, es decir, debe medirse.

3.5.11 Mensajes de error / confirmación y otras indicaciones de condiciones de error

DEFINICIÓN – Mensaje de Error/confirmación
Una Salida emitida por un proceso funcional a un usuario funcional humano que confirma solo que los datos ingresados han sido aceptados, o solo que hay un error en los datos ingresados. NOTA: Cualquier Salida que pueda incluir indicaciones de falla, pero que no esté destinada a un usuario funcional humano, no es un mensaje de error/confirmación.

REGLA – Mensajes de error/confirmación y otras indicaciones de condiciones de error

a) Se identificará una Salida para tener en cuenta todos los tipos de mensajes de error/confirmación emitidos por cualquier proceso funcional del software que se esté midiendo de todas las causas posibles de acuerdo con su FUR, p. ej. éxitos o fracasos de: validación de los datos ingresados o para una llamada para recuperar datos o hacer que los datos sean persistentes, o para la respuesta de un servicio solicitado de otra pieza de software.

   NOTA: Si el FUR del proceso funcional no requiere que se emita ningún tipo de mensaje de error/confirmación, no identifique ninguna Salida correspondiente.

b) Si un mensaje a un usuario funcional humano proporciona datos además de confirmar que los datos ingresados han sido aceptados o que los datos ingresados son erróneos, entonces estos datos adicionales deben identificarse como un grupo de datos movido por una Salida de la manera normal, además de la Salida del error/confirmación.

c) Todos los demás datos, emitidos o recibidos por el software que se está midiendo, a/desde sus usuarios funcionales de hardware o software deben analizarse de acuerdo con el FUR como Salidas o Entradas respectivamente, de acuerdo con las reglas COSMIC normales, independientemente de si los valores de datos indican una condición de error.

d) Las Lecturas y Escrituras se consideran para tener en cuenta, cualquier informe asociado de condiciones de error. Por lo tanto, no se identificará ninguna Entrada al proceso funcional que se esté midiendo para ninguna indicación de error recibida como resultado de una Lectura o escritura de datos persistentes.

e) No se identificará ninguna Entrada o Salida para ningún mensaje que indique una condición de error que podría emitirse mientras se utiliza el software que se está midiendo, pero que no está obligado a ser procesado de ninguna manera por el FUR de ese software, p. ej. Un mensaje de error emitido por el sistema operativo.

EJEMPLO DE NEGOCIO 1 que ilustra la regla a): En un diálogo humano-computadora, los ejemplos de mensajes de error que ocurren durante la validación de los datos ingresados podrían ser “error de formato”, “cliente no encontrado”, “error: marque la casilla de verificación que indica que ha leído nuestros términos y condiciones”, “límite de crédito excedido”, etc. Todos estos mensajes de error deben considerarse como ocurrencias de una Salida en cada proceso funcional donde ocurren dichos mensajes (que podrían denominarse “mensajes de error”).

EJEMPLO DE NEGOCIO 2 que ilustra la regla a): el proceso funcional "A" puede emitir potencialmente 2 mensajes de confirmación distintos y 5 mensajes de error a sus usuarios funcionales. Identifique una Salida para tener en cuenta todos estos mensajes de error/confirmación (5 + 2 = 7). El proceso funcional "B" puede emitir potencialmente 8 mensajes de error a sus usuarios funcionales. Identifique una Salida para dar cuenta de estos 8 mensajes de error.

EJEMPLO DE NEGOCIO 3 que ilustra las reglas a) y b): Un proceso funcional del ATM de un banco (es decir, un “cajero automático” o “dispensador de efectivo”) puede emitir cinco tipos de mensajes en respuesta a una solicitud para retirar una cantidad específica de efectivo:

- *Error: la máquina no tiene efectivo disponible*
- *Error: el monto solicitado debe ser un múltiplo de $ 10*
- *Retiro rechazado. Cuenta bloqueada. Contacta con el banco.*
- *Retiro rechazado (el límite de crédito se excedería en $139.14)*
- *Retiro aceptado; su saldo restante es de $756.25*

Los primeros cuatro mensajes describen una condición de error y la primera parte del quinto mensaje es una confirmación. Para toda esta salida, cuente una Salida, según la regla a). Los dos últimos mensajes también incluyen atributos de datos relacionados con la cuenta del cliente. Cuente una Salida para estos datos, según

la regla b) anterior, y teniendo en cuenta la regla a) de las "Reglas de movimiento de datos únicos" de la sección 3.5.7. En total, identifique dos Salidas para este proceso funcional, es decir, 2 CFP para su salida.

EJEMPLO DE NEGOCIO 4 que ilustra la regla e): Los mensajes de error emitidos a los usuarios humanos, pero no generados o procesados por el software de aplicación que se está midiendo deben ignorarse por completo en la medición de la aplicación. Un ejemplo de dicho mensaje transmitido desde el sistema operativo podría ser "la impresora X no responde".

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

EJEMPLO EN TIEMPO REAL 2 que ilustra la regla c). El FUR del EJEMPLO EN TIEMPO REAL 1 en la sección 3.5.9 también puede indicar que los PF A y B deben manejar una condición de error cuando el software del controlador del dispositivo no puede obtener uno o más datos de la matriz de sensores tontos. Un sensor tonto no puede, por definición, emitir un mensaje de error. El PF B del controlador de dispositivo, muy probablemente, obtendrá una serie de valores de la matriz de sensores tontos, p. ej. estado 1, estado 2, estado 3, sin respuesta, estado 5, sin respuesta, estado 7, etc. y emitirá esta cadena como una Salida al PF A de la aplicación donde se recibe como Entrada. No debe identificarse ningún mensaje de error por separado como una Salida del PF B del software del controlador del dispositivo, ni como una Entrada al PF A de la aplicación de control de procesos.

4

LA FASE DE MEDICIÓN

4.0 Resumen del capitulo

Este capítulo discute el paso final del proceso de medición. Primero, se define la unidad de medida COSMIC (es decir, un movimiento de datos se mide como 1 (un) Punto de Función COSMIC, o "CFP"). Luego se dan las reglas para asignar un tamaño al FUR del software que se está midiendo. Las reglas se definen sobre cómo agregar tamaños de diferentes piezas de software.

Además, las reglas se definen sobre cómo dimensionar los cambios requeridos en el software (como se aborda en los proyectos de "mejora"). Finalmente, el capítulo discute la posibilidad de 'extensiones locales' al método estándar COSMIC que puede usarse, por ejemplo, en el entorno local de una organización que desea dar cuenta de algún aspecto de la funcionalidad de una manera que sea significativa como local estándar.

4.1 El proceso de la fase de medición

El método general para medir una pieza de software cuando sus requisitos de usuario funcional se han expresado en términos del Modelo Genérico de Software COSMIC se resume en la figura 4.0 a continuación.

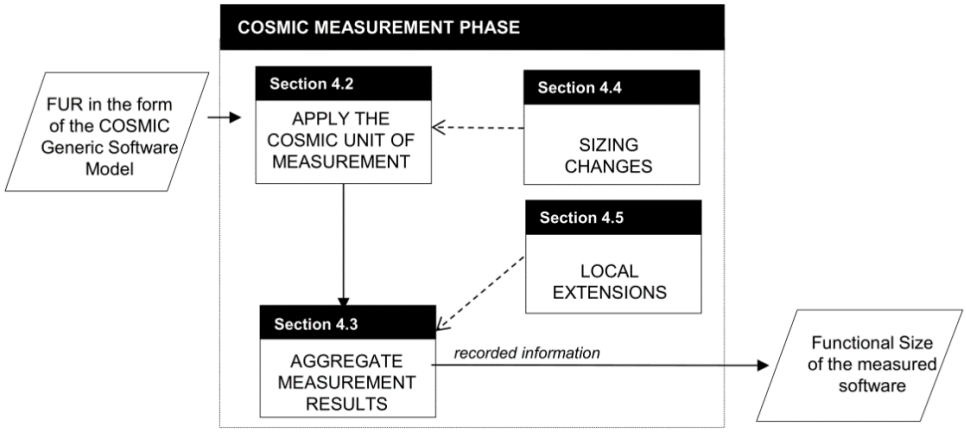


Figura 4.0 - Proceso general para la Fase de Medición COSMIC

Cada paso de este método es el tema 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 (Cosmic Function Point) que se define como el tamaño de un movimiento de datos.
NOTA: La unidad de medida se conocía como "Cfsu" (Unidad de tamaño funcional COSMIC), antes de la v3.0 del método

A partir de esta definición, cada movimiento de datos (Entrada, Salida, Lectura o Escritura) que se requiere agregar, modificar o eliminar para el software que se está midiendo también se mide como 1 CFP.

4.3 Agregando los resultados de medición

Este paso consiste en agregar los tamaños de todos los movimientos de datos identificados, en un solo valor de tamaño funcional. Este paso se realiza de acuerdo con las siguientes reglas.

4.3.1 Reglas generales de agregación

REGLAS – Agregando los resultados de medición

a) Para cualquier proceso funcional, el tamaño funcional de los movimientos de datos individuales se agregará en un solo valor de tamaño funcional en unidades de CFP al sumarlos.

Tamaño (procesos funcionales) = Σ tamaño(Entradas) + Σ tamaño(Salidas) +
Σ tamaño(Lecturas) + Σ tamaño(Escrituras)

b) Para cualquier proceso funcional, el tamaño funcional de los cambios en sus Requisitos funcionales del usuario se agregará a partir del tamaño de los movimientos de datos que se han agregado, modificado o eliminado en el proceso funcional para dar un tamaño del cambio en unidades CFP, de acuerdo con a la siguiente fórmula.

Tamaño (Cambio(procesos funcionales)) = Σ tamaño (mov. de datos agregado) +
Σ tamaño (mov. de datos modificados) +
Σ tamaño (mov. de datos eliminados.)

Para más información sobre agregar tamaño funcional, consulte la sección 4.3.2. Para medir el tamaño del software modificado, consulte la sección 4.4.2

c) El tamaño de una pieza de software dentro de un alcance definido se obtendrá agregando los tamaños de los procesos funcionales para la pieza, sujeto a las reglas e) y f) a continuación.

d) El tamaño de cualquier cambio en una pieza de software dentro de un alcance definido se obtendrá agregando los tamaños de todos los cambios a todos los procesos funcionales para la pieza, sujeto a las reglas e) y f) a continuación

e) Los tamaños de las piezas de software o de los cambios en las piezas de software solo se pueden sumar si se miden al mismo nivel de granularidad de proceso funcional de su FUR.

f) Los tamaños de las piezas de software y/o los cambios en los tamaños de las piezas de software dentro de cualquier capa o de diferentes capas se agregarán juntas solo si tiene sentido hacerlo, para el propósito de la medición.

g) El tamaño de una pieza de software se obtiene sumando los tamaños de sus componentes (independientemente de cómo se descomponga la pieza) y eliminando las contribuciones de tamaño de los movimientos de datos entre componentes.

h) Solo se identificará una Salida para todos los mensajes de error / confirmación emitidos por cualquier proceso funcional a un usuario funcional humano.

i) 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 a través de la extensión local se informará por separado como se describe en la sección 5.1 y NO se agregará al tamaño obtenido por el método estándar, medido en CFP (ver más adelante en la sección 4.5).

EJEMPLO 1 para las reglas b) y c): Un cambio requerido a una pieza de software podría ser: "agregar un nuevo proceso funcional de tamaño 6 CFP, y en otro proceso funcional agregar un movimiento de datos, realizar modificaciones en otros tres movimientos de datos y elimine dos movimientos de datos." El tamaño total del cambio requerido es 6 + 1 + 3 + 2 = 12 CFP.

EJEMPLO 2 para la regla f): si varias partes principales de una pieza de software se desarrollan utilizando diferentes tecnologías, por diferentes sub-equipos de proyecto, puede que no haya un valor práctico al sumar sus tamaños.

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

- *primero medido "como un todo", es decir, todo dentro de un alcance*
- *luego, en segundo lugar, el tamaño de cada uno de sus componentes se mide por separado, es decir, cada uno con su propio alcance,*

entonces el tamaño total de sumar el tamaño de todos los componentes separados (en el segundo caso) excederá el tamaño cuando se mide "como un todo" (en el primer caso) debido a la contribución al tamaño de todos los datos entre componentes movimientos Estos movimientos de datos entre componentes no son visibles cuando la pieza se mide "como un todo". Consulte también el ejemplo en la sección sobre la medición en distintos niveles de granularidad en arquitecturas de software puro en la "Guía para la medición rápida o temprana del tamaño funcional COSMIC" [6])

Cabe señalar que, *dentro* de cada capa identificada, la función de agregación es totalmente escalable. Por lo tanto, se puede generar un subtotal para procesos funcionales individuales o para todo el software dentro de una capa, dependiendo del propósito y el alcance de cada ejercicio de medición y sujeto a las reglas d), e) y f) anteriores.

4.3.2 Más información sobre la agregación de tamaño funcional

En un contexto en el que el tamaño funcional se va a utilizar como una variable en un modelo, por ejemplo, para estimar el esfuerzo, y el software que se va a medir se extiende en más de una capa, la agregación generalmente se realizará por capa, ya que el software en diferentes capas es a menudo no implementado con la misma tecnología.

EJEMPLO 1: Considere el software donde la capa de aplicación se implementará utilizando un lenguaje de tercera generación y un conjunto de bibliotecas existentes, mientras que una capa de controlador se podría implementar utilizando lenguaje ensamblador. El esfuerzo por unidad de tamaño asociado con la construcción del software en cada capa será, muy probablemente, diferente, y, en consecuencia, se preparará una estimación del esfuerzo por separado para el software en cada capa. Es poco probable que haya algún valor en agregar los tamaños del software en las dos capas

EJEMPLO 2: Si un equipo de proyecto tiene que desarrollar una serie de piezas principales de software y está interesado en su productividad general, puede sumar las horas de trabajo necesarias para desarrollar cada pieza. Del mismo modo, el equipo puede sumar los tamaños de las piezas principales que se ha desarrollado si (pero solo si) esos tamaños satisfacen las reglas dadas anteriormente.

La razón por la cual los tamaños de piezas de software de diferentes capas de una arquitectura en capas estándar, medidos con el mismo nivel de granularidad del proceso funcional, se pueden sumar si tiene sentido hacerlo (por ejemplo, todas las piezas de software se desarrollan utilizando la misma tecnología) es que dicha arquitectura tiene un conjunto coherentemente definido de usuarios funcionales. El software en cada capa es un usuario funcional del software en las otras capas que usa y cualquier pieza de software en una capa puede ser un usuario funcional de cualquiera de sus piezas de softwares pares. Por lo tanto, es lógico que los tamaños de las distintas piezas se puedan sumar, siempre que se apliquen las reglas d), e) y f) anteriores. Sin embargo, en contraste, el tamaño de cualquier pieza de software *no* se puede obtener sumando los tamaños de los objetos reutilizables de sus componentes a menos que se eliminen los movimientos de datos entre objetos, según la regla g) anterior.

Agregar los resultados de la medición por tipo de movimiento de datos podría ser útil para analizar la contribución de cada tipo al tamaño total del software en una capa determinada y, por lo tanto, podría ayudar a caracterizar la naturaleza funcional del software medido en la capa dada.

4.4 Más información sobre la medición del tamaño de los cambios en el software

Un "cambio funcional" al software existente se interpreta en el método COSMIC como "cualquier combinación de adiciones de nuevos movimientos de datos o de modificaciones o eliminaciones de movimientos de datos existentes, incluida la manipulación de datos asociada". Los términos "mejora" y "mantenimiento"[12] a menudo se usan para lo que aquí llamamos un "cambio funcional".

La necesidad de un cambio en el software puede surgir de

- un nuevo FUR (es decir, solo adiciones a la funcionalidad existente), o
- de un cambio a la FUR (tal vez involucrando adiciones, modificaciones y eliminaciones) o
- de un "mantenimiento" necesita corregir un defecto

Las reglas para dimensionar cualquiera de estos cambios son las mismas, pero el Medidor recibe una alerta para distinguir las diversas circunstancias al realizar mediciones y estimaciones de rendimiento.

Cuando una pieza de software se reemplaza por completo, por ejemplo, reescribiéndola, con o sin extender y/u omitir la funcionalidad, el tamaño funcional de este cambio es el tamaño del software de reemplazo, medido de acuerdo con las reglas normales para dimensionar un nuevo software. Este caso no se considerará más en esta sección. Sin embargo, el Medidor debe ser consciente de la necesidad al realizar mediciones o estimaciones de rendimiento para distinguir entre proyectos para desarrollar software completamente nuevo y proyectos para "volver a desarrollar" o "reemplazar" el software existente.

A menudo, una parte obsoleta de una aplicación se elimina ("desconectado" sería una mejor descripción) dejando el código del programa en su lugar y simplemente eliminando el enlace a la funcionalidad obsoleta. Cuando la funcionalidad de la parte obsoleta asciende a 100 CFP, pero la parte se puede desconectar cambiando, por ejemplo, 2 movimientos de datos, 100 y no 2 movimientos de datos se identificarán como el tamaño del cambio funcional. Medimos el tamaño del requisito, no el tamaño que se implementó.

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 bastante diferente de las eliminaciones "reales". Alternativamente, para fines de estimación, puede ser preferible medir el tamaño que se implementará (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 CFP, esto debe documentarse claramente y distinguirse de una medición del FUR que requiere que la aplicación se reduzca en tamaño en 100 CFP.

Tenga en cuenta la diferencia entre el tamaño del cambio funcional (discutido aquí) y el cambio en el tamaño funcional del software. Por lo general, son diferentes. En la sección 4.4.2 se aborda el tamaño de este último.

4.4.1 Modificando funcionalidad

Cualquier movimiento de datos de un tipo dado (E, X, R y W) implica dos tipos de funcionalidad: mueve un solo grupo de datos y tiene alguna manipulación de datos asociada (para este último, consulte la sección 3.5.6). Por lo tanto, para fines de medición, se considera que un movimiento de datos se modifica funcionalmente de la siguiente manera.

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

a) Se considera que un movimiento de datos se modifica funcionalmente si se aplica al menos uno de los siguientes:
   - el grupo de datos movido se modifica,
   - se modifica la manipulación de datos asociada.

b) Un grupo de datos se modifica si se aplica al menos uno de los siguientes:
   - uno o más atributos nuevos se agregan al grupo de datos,
   - uno o más atributos existentes se eliminan del grupo de datos,

[12] Una convención de medición normal es que el tamaño funcional de una pieza de software no cambia si el software debe cambiarse para corregir un defecto a fin de alinear el software con su FUR. El tamaño funcional del software cambia si el cambio es para corregir un defecto en el FUR.

- se modifican uno o más atributos existentes, p. ej. en significado o 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 se modifica, por ejemplo, cambiando el cálculo, el formato específico, la presentación y/o la validación de los datos. "Presentación" puede significar, por ejemplo, la fuente, el color de fondo, la longitud del campo, el encabezado del campo, el número de lugares decimales, etc.

Los comandos de control y los datos generales de aplicación de las aplicaciones de negocios no implican movimientos de datos, ya que no se mueven datos sobre objetos de interés. Por lo tanto, los cambios en los comandos de control y los datos generales de la aplicación no deben medirse. Como ejemplo, cuando se cambia el color de la pantalla para todas las pantallas, este cambio no debe medirse. (Consulte la sección 3.5.10 para obtener una explicación de los comandos de control y el Ejemplo empresarial 1 en la sección 3.3.3 para obtener datos generales de la aplicación).

REGLAS – Modificar un movimiento de datos

a) Si un movimiento de datos debe modificarse debido a un cambio en la manipulación de datos asociado con el movimiento de datos y/o debido a un cambio en el número o tipo de los atributos en el grupo de datos movidos, se medirá un CFP modificado, independientemente del número real de modificaciones en un movimiento de datos.

b) Si se debe modificar un grupo de datos, los movimientos de datos que mueven el grupo de datos modificado cuya funcionalidad no se ve afectada por la modificación del grupo de datos no se identificarán como movimientos de datos modificados.

NOTA: Una modificación a cualquier dato que aparezca en las pantallas de entrada o salida que no estén relacionadas con un objeto de interés para un usuario funcional no se identificará como un CFP modificado. (Consulte la sección 3.3.3 para ver ejemplos de dichos datos).

EJEMPLO para la regla a): Una solicitud de cambio para un proceso funcional requiere tres cambios en la manipulación de datos asociada con su Entrada desencadenante y dos cambios en la manipulación asociada con una Salida, así como dos cambios en los atributos del grupo de datos movido por este Salida. Mida el tamaño del cambio como 2 CFP, es decir, cuente el número total de movimientos de datos cuyos atributos y manipulación de datos asociados deben cambiarse. NO cuente la cantidad de manipulaciones de datos o atributos de datos que se van a cambiar.

EJEMPLO para las reglas a) y b): Suponga un requisito para agregar o modificar los atributos de datos de un grupo de datos D_1, de modo que después de la modificación se convierta en D_2. En el proceso funcional A donde se requiere esta modificación, todos los movimientos de datos afectados por la modificación deben identificarse y contarse como modificados. Entonces, según la regla a), si el grupo de datos modificado D_2 se hace persistente y/o se emite en el proceso funcional A, identifique un movimiento de datos de Escritura y/o de Salida, respectivamente, como modificado. Sin embargo, es posible que otros procesos funcionales lean o ingresen este mismo grupo de datos D_2, pero su funcionalidad no se ve afectada por la modificación porque no procesan los atributos de datos modificados o agregados. Estos procesos funcionales continúan procesando el grupo de datos movido como si todavía fuera D_1. Entonces, según la regla (b), estos movimientos de datos en los otros procesos funcionales que no se ven afectados por la modificación de los movimientos de datos del proceso funcional A NO deben identificarse y contarse como modificados.

EJEMPLO DE NEGOCIOS: si se requiere cambiar un mensaje de error/confirmación (es decir, textos agregados, modificados o eliminados), debe identificarse para la medición, independientemente de si el texto modificado es o no una consecuencia de un requisito para cambiar otro movimiento de datos.

4.4.2 Tamaño del software funcionalmente modificado

REGLAS – Tamaño del software funcionalmente modificado

Después de cambiar funcionalmente una pieza de software:

Nuevo tamaño total (pieza de software modificada) = Tamaño total anterior (pieza de software)
+ Σ tamaño (movimientos de datos agregados)
- Σ tamaño (movimientos de datos eliminados)

Los movimientos de datos modificados no influyen en el tamaño de la pieza de software, ya que existen tanto antes como después de que se hayan realizado las modificaciones.

EJEMPLO: Recuerde el Ejemplo 1 en la sección 4.3.1: "Un cambio requerido a una pieza de software podría ser: 'agregue un nuevo proceso funcional de tamaño 6 CFP, y en otro proceso funcional agregue un movimiento de datos, realice modificaciones en otros tres movimientos de datos y elimine dos movimientos de datos. 'El tamaño total del cambio requerido es 6 + 1 + 3 + 2 = 12 CFP ".

El tamaño total de la pieza de software habrá aumentado en 6 CFP debido a la adición del nuevo proceso funcional y habrá disminuido en 1 CFP debido a las adiciones a (+1 CFP) y las eliminaciones de (-2 CFP) otro proceso funcional. Después del cambio, la pieza de software habrá aumentado de tamaño en (+6 - 1) = 5 CFP.

4.5 Extendiendo el método de medición COSMIC

4.5.1 Introducción

El método COSMIC para medir un tamaño funcional no supone medir todos los aspectos posibles del "tamaño" del software. Por lo tanto, el método no está diseñado actualmente para medir por separado y explícitamente el tamaño del FUR de los subprocesos de manipulación de datos. La influencia en el tamaño de los subprocesos de manipulación de datos se tiene en cuenta a través de un supuesto simplificador que es válido para una amplia gama de dominios de software, como se define en la sección 1.1 sobre la aplicabilidad del método. Además, no se captura la influencia del número de atributos de datos por movimiento de datos en el tamaño del software.

Se puede considerar que otros parámetros, como la "complejidad" (según se defina) contribuyen al tamaño funcional. Un debate constructivo sobre este asunto requeriría primero definiciones comúnmente acordadas de los otros elementos dentro de la noción mal definida de "tamaño" tal como se aplica al software. Estas definiciones siguen siendo, en este punto, el tema de una mayor investigación y de mucho debate.

Sin embargo, la medida de tamaño COSMIC se considera una buena aproximación para el propósito declarado del método y los dominios de aplicabilidad. Sin embargo, puede ser que dentro del entorno local de una organización que utiliza el método de medición COSMIC, se desee tener en cuenta dicha funcionalidad de una manera que sea significativa como estándar local. Por esta razón, el método de medición COSMIC tiene disposiciones para extensiones locales. Cuando se usan tales extensiones locales, los resultados de la medición deben informarse de acuerdo con la convención especial presentada en la sección 5.1. Las siguientes secciones muestran cómo extender el método con un estándar local.

4.5.2 Software rico en manipulación de dato.

El método COSMIC fue diseñado para medir el software "rico en movimiento de datos". Al igual que todos los demás métodos verdaderos de medición funcional del tamaño (FSM), no fue diseñado para medir explícitamente la funcionalidad de la manipulación de datos. En cambio, el método supone que los tipos de movimiento de datos representan la funcionalidad de manipulación de datos asociada (ver más abajo). Esta suposición ha demostrado ser razonable para todos los fines prácticos, como la medición del desempeño del proyecto y la estimación para la cual se diseñó el método y para los dominios en los que se usa comúnmente.

Sin embargo, la experiencia ha demostrado que el método a menudo también puede aplicarse con éxito al tamaño del software "rico en manipulación de datos", p. ej. algún software científico/de ingeniería. Esto es cierto, por ejemplo, cuando el software debe manejar grandes volúmenes de datos, lo que lleva a un gran número de tipos de movimiento de datos. Este último puede explicar efectivamente cualquier manipulación de datos matemáticamente complejas que también puede estar presente. Por "aplicado con éxito", queremos decir que el método ha producido tamaños significativos y útiles en relación con el propósito de la medición. Los ejemplos

incluyen el dimensionamiento de sistemas expertos, software para procesar digitalmente variables continuas, software que recolecta y analiza datos de experimentos científicos o de mediciones de ingeniería, etc.

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

4.5.3 Limitaciones de los factores que contribuyen al tamaño funcional

Dentro de sus dominios de aplicabilidad, el método COSMIC no intenta medir todos los aspectos posibles de la funcionalidad que podrían considerarse que contribuyen al "tamaño" del software. Por ejemplo, el método de medición no captura explícitamente la influencia de la "complejidad" del software. Pero hay muchos tipos de complejidad, p. arquitectónico, semántico, de tiempo, proceso, datos, etc., y al medir el tamaño funcional, el método en realidad explica de manera simple la contribución al tamaño de la complejidad del proceso (y, por lo tanto, indirectamente de la complejidad de los datos).

El método tampoco considera la influencia del número de atributos de datos por movimiento de datos en el tamaño funcional del software. Como se describe en la sección 4.5.6, si se desea, dichos aspectos del tamaño funcional pueden estar respaldados por una extensión local del método de medición COSMIC.

4.5.4 Limitaciones en la medición de piezas de software muy pequeñas

Todos los métodos de medición del tamaño funcional se basan en los supuestos de un modelo simplificado de funcionalidad de software que pretende ser razonable "en promedio" para su dominio de aplicabilidad y usos en la medición y estimación del rendimiento del proyecto. Por lo tanto, se necesita precaución al medir, comparar o usar tamaños de piezas de software muy pequeñas para estos fines, y especialmente de cambios muy pequeños en una pieza de software, donde la suposición de "promedio" puede romperse. En el caso del método COSMIC, "muy pequeño" significa "algunos movimientos de datos".

Nota: los consejos anteriores de "precaución" no deben evitar el uso del método COSMIC en las actividades de software ágil. Por el contrario, el método se está utilizando con éxito para medir el tamaño de Historias de Usuarios en desarrollos ágiles de software, que pueden tener muy pocos movimientos de datos. Esta práctica está bien establecida, con muchos informes de los tamaños en CFP de las iteraciones ágiles (o “sprints”), agregados a partir de los tamaños de las Historias de Usuarios individuales, que se correlacionan muy bien con el esfuerzo por desarrollar la iteración (y mucho mejor que la correlación con el esfuerzo de tamaños medidos usando Puntos de Historia). Para un informe de ejemplo sobre tal comparación, ver [16].

4.5.5 Extensión local con algoritmos complejos.

Si se considera necesario dar cuenta de algoritmos complejos, se puede organizar un estándar local para esta funcionalidad excepcional. En cualquier proceso funcional donde haya un subproceso funcional de manipulación de datos anormalmente complejo, el Medidor es libre de asignar sus propios Puntos de Función determinados localmente.

> *EJEMPLO: Un estándar de extensión local podría ser: ‘En nuestra organización, se asigna un Punto de Función local para algoritmos matemáticos tales como (lista de ejemplos localmente significativos y bien entendidos). Se asignan dos Puntos de Función locales para (otra lista de ejemplos), etc."*

4.5.6 Extensión local con subunidades de medida

Cuando se requiere más precisión en la medición de los movimientos de datos, se puede definir una subunidad de la medida. Por ejemplo, un medidor puede subdividirse en 100 centímetros o 1000 milímetros. Por analogía, el movimiento de un solo atributo de datos podría usarse como una subunidad de medida. Las mediciones en una pequeña muestra de software en las pruebas de campo de COSMIC indicaron que, en el software medido, el número promedio de atributos de datos por movimiento de datos no varió mucho entre los cuatro tipos de movimiento de datos. Por esta razón y por razones de facilidad de medición, la unidad de medición COSMIC, 1 CFP, se ha fijado al nivel de un movimiento de datos. Sin embargo, se necesita precaución cuando se comparan los tamaños medidos en CFP de dos piezas diferentes de software donde el número promedio de atributos de datos por movimiento de datos difiere considerablemente entre las dos piezas de software.

Cualquier persona que desee refinar el método COSMIC mediante la introducción de una subunidad de medida es libre de hacerlo, pero debe dejar en claro que las medidas de tamaño resultantes no se expresan en puntos de función COSMIC estándar.

5

REPORTE DE MEDICIÓN

5.0 Resumen del Capitulo

Cuando una medición es finaliza y se acepta, el resultado debe informarse y los datos sobre la medición deben archivarse para garantizar que el resultado sea siempre inequívocamente interpretable. El capítulo enumera los parámetros que deben considerarse para la grabación.

5.1 Etiquetado

El Modelo Genérico de Software se puede representar en forma de matriz donde las filas representan los procesos funcionales (que pueden estar agrupados por capas), las columnas representan grupos de datos y las celdas contienen los subprocesos identificados (entrada, salida, lectura y escritura). Esta representación del modelo de software genérico se presenta en el apéndice A.

Los resultados de las mediciones COSMIC se deben informar y archivar de acuerdo con las siguientes convenciones. Al informar un tamaño funcional COSMIC, debe etiquetarse de acuerdo con la siguiente convención, de acuerdo con la norma ISO / IEC 14143-1: 2007.

REGLA – Etiquetado de la medición COSMIC

El resultado de una medición COSMIC se anotará como ‘x CFP (v)‘, donde:

- "x" representa el valor numérico del tamaño funcional,
- "v" representa la identificación de la versión del método COSMIC estándar utilizado para obtener el valor numérico de tamaño funcional "x"

NOTA: Si se usó un método de aproximación local para obtener la medición, pero de lo contrario la medición se realizó utilizando las convenciones de una versión COSMIC estándar, se utilizará la convención de etiquetado anterior, pero el uso del método de aproximación se debe anotar en otra parte - vea la sección 5.2.

EJEMPLO: Un resultado obtenido usando las reglas de este Manual de Medición se anota como "x CFP (v4.0.2)"

Cuando se utilizan extensiones locales, como se define en la sección 4.5 anterior, el resultado de la medición debe informarse como se define a continuación.

REGLA – Etiquetado de extensiones locales COSMIC

El resultado de una medición COSMIC usando extensiones locales se anotará como:

“**x** CFP (v.) + Z Local FP”, donde:

- "**x**" representa el valor numérico obtenido al agregar todos los resultados de medición individuales de acuerdo con el método COSMIC estándar, versión v,
- "v" representa la identificación de la versión del método COSMIC estándar utilizado para obtener el valor numérico de tamaño funcional "x"
- "z" representa el valor numérico obtenido al agregar todos los resultados de medición individuales obtenidos de extensiones locales al método COSMIC.

5.2 Archivado de resultados de mediciones COSMIC

Al archivar resultados de mediciones COSMIC, la siguiente información debe mantenerse para garantizar que el resultado sea siempre interpretable.

REGLA – Informe de mediciones COSMIC

Además de las mediciones reales, registradas como en 5.1, se deben registrar algunos o todos los siguientes atributos de cada medición, dependiendo del propósito de la medición y el nivel deseado de comparabilidad con otras mediciones, p. para fines de evaluación comparativa.

a) Identificación del componente de software medido (nombre, ID de versión o ID de configuración).

b) Las fuentes de información utilizadas para identificar el FUR utilizado para la medición.

c) El dominio del software.

d) Una descripción de la arquitectura de las capas en las que se realiza la medición, si corresponde.

e) Una declaración del propósito de la medición.

f) Una descripción del alcance de la medición, y su relación con el alcance general de un conjunto relacionado de mediciones, si corresponde. (Utilice las categorías de alcance genérico en la sección 2.2).

g) El patrón de medición (estrategia) utilizado (COSMIC o local), con el modo de procesamiento (en línea 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 realizó la medición (especialmente si la medición es una estimación basada en requisitos incompletos, o se realizó sobre la base de la funcionalidad realmente entregada).

k) El margen de error objetivo o creído de la medición.

l) Indica si se usó el método de medición COSMIC estándar, y/o una aproximación local al método estándar, y/o si se usaron extensiones locales (ver sección 4.5). Utilice las convenciones de etiquetado de las secciones 5.1 o 5.2.

m) Una indicación de si la medición es de funcionalidad desarrollada o entregada (la funcionalidad 'desarrollada' se obtiene mediante la creación de un nuevo software; la funcionalidad 'entregada' incluye la funcionalidad 'desarrollada' y también incluye la funcionalidad obtenida por otros medios que no sean la creación de un nuevo software, es decir, incluyendo todas las formas de reutilización de software existente, implementación de paquetes de software, uso de parámetros existentes para agregar o cambiar funcionalidades, etc.

n) Una indicación de si la medición es de una funcionalidad recientemente proporcionada o es el resultado de una actividad de "mejora" (es decir, la suma es de funcionalidad agregada, modificada y eliminada - ver 4.4).

o) El número de componentes principales, si corresponde, cuyos tamaños se han sumado para el tamaño total registrado.

p) El porcentaje de funcionalidad implementada por el software reutilizado.

q) Para cada alcance dentro del alcance de medición general, una matriz de medición, como se especifica en el Apéndice A.

r) El nombre del Medidor y cualquier calificación de certificación COSMIC; La fecha de la medición.

Referencia

Todos los documentos COSMIC enumerados a continuación, incluidas las traducciones a otros idiomas, se pueden encontrar en www.cosmic-sizing.org.

Los títulos de los documentos COSMIC no dan el número de versión del método al que se refieren. Todos los documentos se actualizan periódicamente para alinearlos con la última versión del método.

[1] ISO/IEC 19761:2017 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 NonFunctional 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 early or rapid COSMIC functional size measurement by using approximation approaches

[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] Guideline on how to convert 'First Generation' Function Point sizes to COSMIC sizes

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

[14] Adapted from Merriam Webster's Collegiate Dictionary, 10 th Edition

[15] Adapted from Merriam Webster's Collegiate Dictionary, 10 th Edition, and La Petit Larousse Illustré, 1996 Edition

[16] Effort Estimation with Story Points and COSMIC Function Points – An Industry Case Study, Christophe Commeyne, Alain Abran, Rachida Djouab, Software Measurement News, Vol 21, No. 1 2016.

[17] ISO/IEC 14143/1:2011 Information technology – software measurement – functional size measurement. Part 1 Definition of concepts

[18] Glossary of terms for 'Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating', COSMIC and IFPUG, September 2015.

[19] See for example: www.wikipedia.org/wiki/AUTOSAR

APENDICE A – DOCUMENTANDO EL TAMAÑO DE LA MEDICIÓN COSMIC

La estructura a continuación se puede usar como un repositorio para guardar los resultados de una medición para cada elemento de software identificado de un alcance general que se ha asignado al Modelo Genérico de Software. Cada alcance dentro del alcance de medición general tiene su propia matriz.

La base de conocimiento de www.cosmic-sizing.org tiene una serie de herramientas de hoja de cálculo para el registro de mediciones.

Layer Name															
		Data Group Names													
	Software Name A	Data Group 1	...	...	...	...	...	...	Data Group n	Entries	Exits	Reads	Writes	Total
	Functional Process 1													
	Functional Process 2													
	Functional Process 3													
	Functional Process 4													
	Functional Process 5													
						Software A Totals								
Layer Name														
		Data Group Names													
	Software Name B	Data Group 1	...	...	...	...	...	...	Data Group n	Entries	Exits	Reads	Writes	Total
	Functional Process 1													
	Functional Process 2													
	Functional Process 3													
						Software B Totals								
					Total Software A + B									

Figura A – Matriz del Modelo Genérico de Software

FASE DE ESTRATEGIA DE MEDICIÓN

- Cada pieza de software con un alcance definido en cada capa se puede registrar como un elemento de software separado

FASE DE MAPEO

- Cada grupo de datos identificado se registra en una columna.
- Cada proceso funcional se registra en una línea específica, agrupada por un elemento de software identificado.
- Para cada proceso funcional identificado, los movimientos de datos identificados ya sean nuevos o modificados, se anotan en la celda correspondiente utilizando la siguiente convención: 'E' para una Entrada, 'X' para una Salida, 'R' para una Lectura y 'W 'para una Escritura

FASE DE MEDICIÓN

- 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 la medición se puede calcular y registrar en las celdas encuadradas de cada componente, en la línea "TOTAL".

APENDICE B - EVOLUCIÓN DE REQUISITOS NO FUNCIONALES - EJEMPLOS

La siguiente tabla enumera algunos ejemplos de declaraciones de requisitos que pueden aparecer inicialmente a nivel del sistema (incluso antes de que los requisitos se hayan asignado al software o hardware) o al nivel de software como no funcionales pero que evolucionan, total o parcialmente a medida que avanza un proyecto, en una mezcla de FUR para software y declaraciones de requisitos que son realmente “no funcionales”.

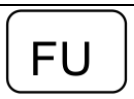
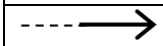
- Columna 1: ejemplos de declaraciones de sistema o software NFR
- Columna 2: ejemplos de FUR de software que podrían resultar, a medida que avanza un proyecto, del NFR en la columna 1. El FUR puede ser para desarrollar o adquirir software, p. ej. Software "COTS" (Comercial fuera de la plataforma).
- Columna 3: ejemplos de los requisitos y restricciones en el sistema o proyecto que pueden quedar después de separar el FUR del software en la columna 2. Por lo tanto, estos son requisitos "no funcionales" verdaderos.

Requisitos del sistema o software que pueden aparecer inicialmente como no funcionales	**Ejemplos de FUR para software, para ser desarrollado o adquirido, que puede evolucionar desde el sistema inicial NFR**	**Ejemplos de NFR verdadero que pueden permanecer después de que algunos requisitos iniciales del sistema se hayan convertido en software FUR**
El tiempo de respuesta del sistema durante la hora pico no debe exceder un promedio de X segundos.	Software para: • Alimentar los datos externos que necesita el sistema en tiempo real. • Monitorear e informar sobre el tiempo promedio de respuesta.	• Hardware específico (rápido) • Algún software para escribir en un lenguaje de bajo nivel. • La declaración de objetivo de tiempo de respuesta específico
La disponibilidad del sistema excederá el Y% promedio durante cada año calendario	Software para permitir el cambio rápido de procesamiento a un procesador de respaldo sin interrupción al servicio	• Procesador de hardware de respaldo que funciona en modo de "espera activa" • La declaración de objetivo de disponibilidad específica
Los parámetros de la aplicación serán fácilmente mantenibles por el personal del usuario	Software para permitir a los usuarios mantener tablas de parámetros	(Ninguno)
El sistema podrá ser utilizado por miembros del público en general sin capacitación y con una tasa de finalización exitosa del Z%	Software para: • proporcionar instalaciones de ayuda integrales • proporcionar menús bien estructurados para facilitar su uso. • apoyar a usuarios con visión parcial	• Requisitos para teclados Braille • Pruebas exhaustivas por parte del público en general. • La declaración de tasa de finalización de objetivo Z% específica
El usuario tendrá la opción de proteger los archivos mediante encriptación	Software para cifrar y descifrar archivos a pedido del usuario	Uso de un "dongle" de hardware o dispositivo de clave de cifrado
El sistema será portátil en entornos de hardware / software X, Y y Z	Una capa de software para aislar la funcionalidad principal de los requisitos de interfaz específicos de los entornos X, Y y Z	Uso de un lenguaje altamente portátil como Java

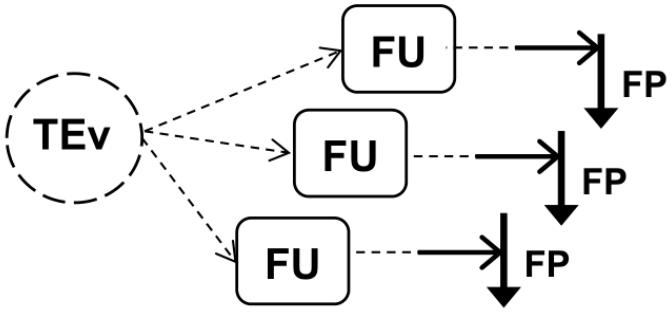
APENDICE C - CARDINALIDAD DE EVENTOS DE DISPARO, USUARIOS Y PROCESOS FUNCIONALES

Todas las relaciones a lo largo del evento desencadenante/usuario funcional/Entrada desencadenante/cadena de proceso funcional (como se muestra en la Figura 3.3) pueden ser de principio a fin, con una excepción. (La excepción es que cualquiera Entrada desencadenante puede iniciar solo un proceso funcional; consulte 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 usa las siguientes abreviaturas y símbolos

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

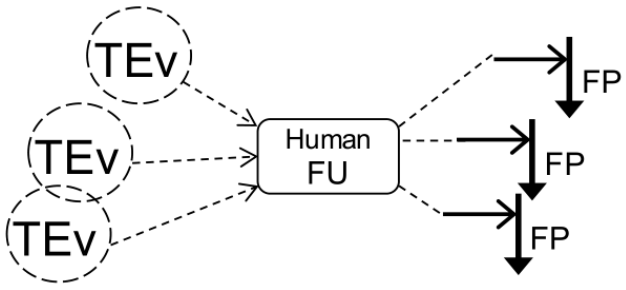
1. Un solo evento desencadenante puede causar que varios FU inicien una Entrada desencadenante en el mismo o en diferentes sistemas de software. Cada Entrada desencadenante comienza su propio PF.



EJEMPLO EN TIEMPO REAL: El evento desencadenante de un terremoto puede ser detectado por múltiples sensores FU independientes. Cada FU inicia una Entrada desencadenante que inicia su FP en el mismo sistema o en sistemas diferentes.

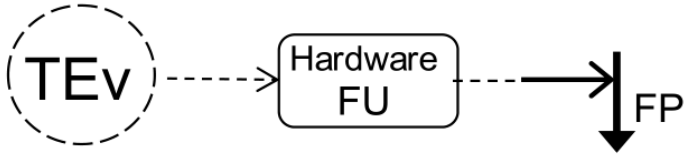
EJEMPLO DE NEGOCIO: El evento desencadenante de un nuevo empleado que comienza a trabajar hace que un FU humana ingrese datos básicos de empleados a un sistema de Personal y otro FU humana ingrese datos de salario a un sistema de Nómina.

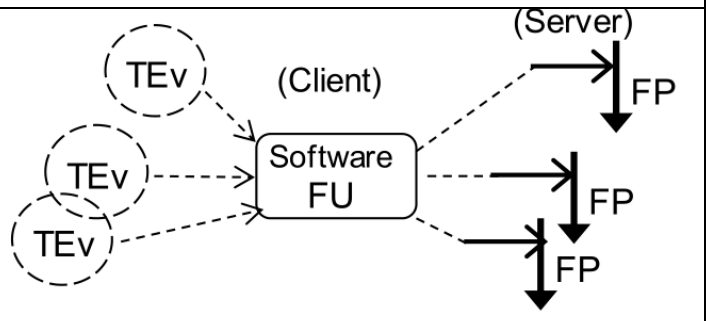
2. Cada evento desencadenante hace que un FU humano inicie una Entrada desencadenante diferente. Cada Entrada desencadenante inicia su FP en el mismo o en diferentes sistemas de software.



EJEMPLO DE NEGOCIO: En un sistema de manejo de llamadas telefónicas de emergencia de la policía, se pueden informar muchos tipos de eventos desencadenantes que provocan que un UF humano decida iniciar diferentes Entradas desencadenantes. Cada uno de estos comienza su FP para grabar el evento. Además, el usuario humano puede iniciar Entradas desencadenantes de consulta diferentes. Cada uno de estos comienza su FP en el mismo sistema de manejo de llamadas o en otros sistemas.

3. Un FU de hardware o software puede estar diseñado para detectar (o "generar") uno o más tipos específicos de eventos. Cada uno de estos hace que la FU inicie una Entrada desencadenante. Cada uno de estos inicia su FP en el mismo sistema de software.

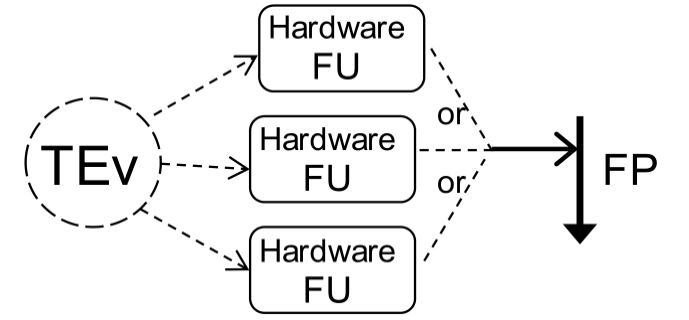




EJEMPLO EN TIEMPO REAL: Cuando la temperatura de un líquido alcanza un nivel preestablecido (el evento desencadenante), un termopar FU inicia una Entrada desencadenante para iniciar su FP en un sistema de software específico.

EJEMPLO DE NEGOCIO: En una aplicación de software distribuido, el componente del cliente es un FU del componente del servidor. Las diferentes necesidades de información (los eventos desencadenantes) del componente del cliente hacen que inicie diferentes Entradas desencadenantes, cada una para iniciar su FP del componente del servidor, para cada tipo diferente de servicio que necesita.

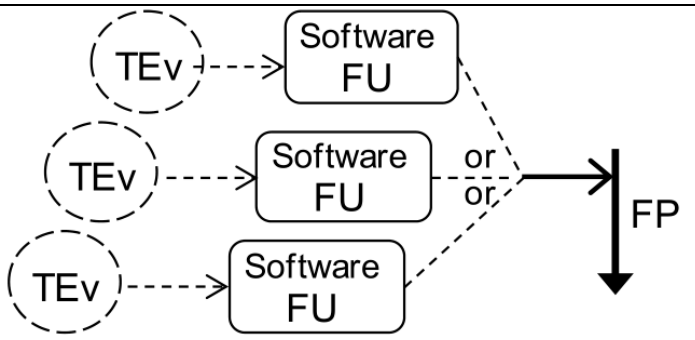
4. Dos o más FU de hardware del mismo software pueden detectar el mismo evento desencadenante. Cada FU puede iniciar la Entrada desencadenante que inicia el mismo FP.



EJEMPLO EN TIEMPO REAL: El evento desencadenante de una situación anormal en un sistema de control de proceso en tiempo real puede ser detectado por una o más unidades de hardware. Cada FU puede iniciar un FP de apagado de emergencia.

(NOTA: Cualquier ocurrencia de este FP será iniciada por el primer FU para detectar el evento desencadenante).

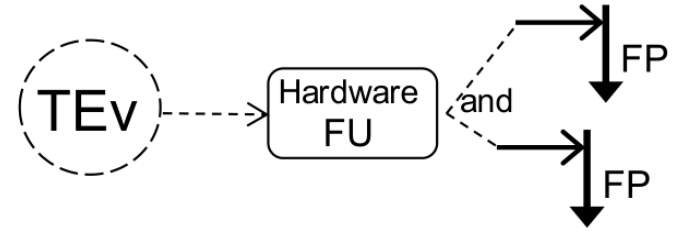
5. Dos o más FU de software pueden iniciar una Entrada desencadenante que inicie el mismo FP.



EJEMPLO DE INFRAESTRUCTURA: Varios UF de software pueden cada una 'llamar', es decir, iniciar el mismo FP en el mismo componente de software reutilizable. (En este caso, el software FU "genera" el evento cuando llama al componente).

(NOTA: Cualquier aparición de este FP puede ser iniciada por solo una de sus posibles FU de software en cualquier momento).

6. Al detectar un evento desencadenante, un UF puede iniciar dos o más Entradas desencadenantes. Cada Entrada desencadenante comienza su FP.



EJEMPLO EN TIEMPO REAL: en un sistema de control dúplex crítico para la seguridad, un evento desencadenante puede hacer que un FU (generalmente hardware) inicie dos Entradas desencadenantes, cada una de las cuales inicia su FP. Los dos PF podrían, por ejemplo, tener el mismo FUR, pero ser desarrollados por grupos separados como resultado de una estrategia de diversidad.

APENDICE D - RESUMEN DE PRINCIPIOS Y NORMAS DEL MÉTODO CÓSMICO

La siguiente tabla identifica cada principio y regla que se encuentra en el Método de medición COSMIC v4.0.2, con el fin de hacer una referencia precisa, con el número de sección en la columna izquierda.

Sec.	DESCRIPCIÓN DE PRINCIPIOS Y REGLAS
1.3.1	**El Modelo Contextual de Software COSMIC** **Principios** a) El software está limitado por el hardware b) El software esta típicamente estructurado en **capas** c) Una capa puede tener una o más piezas de software "**similares**" separadas d) Cualquier software que deba medirse, se definirá por su **alcance de medición**, que se limitará completamente dentro de una sola capa e) El alcance de una pieza de software a medir dependerá del **propósito** de la medición f) Los **usuarios funcionales** de una pieza de software que se medirá se identificarán a partir de sus **Requisitos de Usuario Funcional** (FUR) como remitentes y/o destinatarios de datos deseados a/desde el software respectivamente. g) Los requisitos funcionales del software pueden expresarse a diferentes **niveles de granularidad.** h) Una medida de tamaño COSMIC precisa de una pieza de software requiere que sus FUR sean conocidos en los niveles de granularidad en los que se pueden identificar sus **procesos funcionales** y subprocesos. i) Si los requisitos funcionales de una pieza de software están disponibles solo en un alto nivel de granularidad, se puede utilizar un enfoque de aproximación para medir un tamaño en el alto nivel de granularidad y para escalar el resultado a un tamaño COSMIC aproximado a los niveles del Procesos funcionales y movimientos de datos.
1.3.2	**El Modelo Genérico de Software** **Principios** a) Una pieza de software interactúa con sus usuarios funcionales a través de una **frontera** y con un **almacenamiento persistente** dentro de este límite. b) Los requisitos del usuario funcional de una pieza de software a medir se pueden mapear en procesos funcionales únicos. c) Cada proceso funcional consiste en subprocesos. d) Un subproceso puede ser un **movimiento de datos** o una manipulación de datos. e) Un movimiento de datos mueve un solo **grupo de datos**. f) Hay cuatro tipos de movimiento de datos, **Entrada, Salida, Escritura** y **Lectura**. • Una Entrada mueve un grupo de datos a un proceso funcional desde un usuario funcional. • Una Salida saca un grupo de datos de un proceso funcional a un usuario funcional. • Una Escritura mueve un grupo de datos de un proceso funcional a un almacenamiento persistente.

	• Una Lectura mueve un grupo de datos del almacenamiento persistente a un proceso funcional g) Un grupo de datos consta de un conjunto único de **atributos de datos** que describen un único objeto de interés. h) Cada proceso funcional se inicia por su movimiento de datos de **Entrada desencadenante**. El grupo de datos movido por la Entrada desencadenante es generado por un usuario funcional en respuesta a un **evento desencadenante**. i) El tamaño de un proceso funcional es igual a la cuenta total de sus movimientos de datos. j) Un proceso funcional debe incluir al menos el movimiento de datos de Entrada desencadenante y un movimiento de datos de Escritura o de Salida, es decir, debe incluir un mínimo de dos movimientos de datos. No hay un límite superior para el número de movimientos de datos en un proceso funcional y, por lo tanto, no hay un límite superior para su tamaño. k) Como una aproximación para fines de medición, los subprocesos de manipulación de datos no se miden por separado; se supone que la funcionalidad de cualquier manipulación de datos se debe al movimiento de datos con el que está asociada. NOTA: El Modelo Genérico de Software de COSMIC, como su nombre lo indica, es un "modelo" lógico que expone las unidades en las que el software procesa datos que son adecuados para la medición de tamaño funcional. El modelo no pretende describir la secuencia física de los pasos por los que se ejecuta el software ni ninguna implementación técnica del software.
1.4	**El principio de medición COSMIC** **Principios** a) El tamaño de un proceso funcional es igual al número de sus movimientos de datos. b) El tamaño funcional de una pieza de software de alcance definido es igual a la suma de los tamaños de sus procesos funcionales.
2.2	**Alcance de la Medición** **Reglas** a) El alcance de cualquier pieza de software a medir se derivará del propósito de la medición. b) El alcance de cualquier medición no debe extenderse a más de una capa del software a medir.
2.2.2	**Capas** **Principios** a) El software en una capa proporciona un conjunto de servicios que es cohesivo según un criterio definido, y que el software en otras capas puede utilizar sin saber cómo se implementan esos servicios. b) La relación entre el software en cualquiera de las dos capas se define por una "regla de correspondencia" que puede ser • "jerárquica", es decir, el software en la capa A puede usar 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 puede usar software en la capa B y viceversa. c) El software en una capa intercambia los grupos de datos con el software en otra capa a través de sus respectivos procesos funcionales. d) El software en una capa no usa necesariamente todos los servicios funcionales suministrados por el software en otra capa. e) El software en una capa de una arquitectura de software definida puede dividirse en otras capas de acuerdo con una arquitectura de software definida diferente.

2.3.1	**Usuarios Funcionales** **Reglas** a) Los usuarios funcionales de una pieza de software a medir dependerán del propósito de la medición. b) Cuando el propósito de una medición de una pieza de software se relaciona con el esfuerzo por desarrollar o modificar la pieza de software, los usuarios funcionales deben ser todos los diferentes tipos de remitentes y/o destinatarios de datos deseados a/desde la funcionalidad nueva o modificada, según lo requerido por su FUR. NOTA: FUR puede especificar que múltiples apariciones de usuarios funcionales deben identificarse individualmente. Sin embargo, serán del mismo tipo si cada aparición está sujeta a la misma FUR.
2.4.3	**Niveles de granularidad para medir un proceso funcional** **Reglas** a) Una medición de tamaño funcional de una pieza de software requiere que sus FUR sean conocidos en los niveles de granularidad en los que se pueden identificar sus procesos funcionales y sus subprocesos 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 pueden medirse utilizando un enfoque aproximado. Estos enfoques definen cómo los requisitos pueden medirse a niveles más altos de granularidad. Luego, estos factores de escala se aplican a las mediciones en los niveles más altos de granularidad para producir un tamaño aproximado en los niveles de granularidad de los procesos funcionales y sus subprocesos de movimiento de datos. Consulte la “Guía para la medición temprana o rápida por un enfoque de aproximación del tamaño funcional COSMIC.” [6]
3.2.2	**Procesos Funcionales** **Reglas** a) Un proceso funcional pertenecerá íntegramente al alcance de la medición de una pieza de software en una, y sólo una, capa. b) Un proceso funcional debe comprender un mínimo de dos movimientos de datos, es decir, la Entrada desencadenante más una Salida o una Escritura, dando un tamaño mínimo de 2 CFP. No hay límite superior para el número de movimientos de datos en un proceso funcional y, por lo tanto, no hay límite superior para su tamaño. c) Un proceso funcional en ejecución se considerará terminado cuando haya cumplido con su FUR para todas las respuestas posibles a su Entrada desencadenante. Una pausa durante el procesamiento por razones técnicas no se considerará como finalización del proceso funcional.
3.3.1	**Grupo de Datos** **Principios** Cada grupo de datos identificado será único y distinguible a través de su colección única de atributos de datos.
3.3.2	**Identificar diferentes grupos de datos (y, por lo tanto, diferentes objetos de interés) movidos en el mismo proceso funcional** **Reglas** Para todos los atributos de datos que aparecen en la entrada de un proceso funcional: a) los conjuntos de atributos de datos que tienen diferentes frecuencias de ocurrencia describen diferentes objetos de interés; b) conjuntos de atributos de datos que tienen la misma frecuencia de ocurrencias, pero diferentes atributos clave de identificación describen diferentes objetos de interés;

	c) todos los atributos de datos en un conjunto que resultan de aplicar las partes a) y b) de esta regla pertenecen al mismo *tipo* de grupo de datos, a menos que el FUR especifique que puede haber más de un tipo de grupo de datos que describa el mismo objeto de interés en la entrada al proceso funcional (ver Nota 3) Esta misma regla se aplica a todos los atributos de datos que aparecen en la salida de un proceso funcional, o todos los que se mueven de un proceso funcional a un almacenamiento persistente, o todos los que se mueven de un almacenamiento persistente a un proceso funcional. NOTA 1. Puede ser útil al analizar resultados complejos, por ejemplo, informes con datos que describen varios objetos de interés, para considerar cada grupo de datos candidato separado como si fuera un proceso funcional separado. Cada uno de los tipos de grupos de datos identificados de esta manera también debe distinguirse y contarse al medir el informe complejo. Para ver ejemplos, consulte la “Guía de Medición para las Aplicaciones de Negocios” [7], en particular el ejemplo en la sección 2.6.1 y su análisis en 2.6.2. Ver también el análisis de los ejemplos 4 y 5 en la sección 4.2.4. NOTA 2. Examinar cómo los atributos de datos se agrupan o se separan físicamente en la *entrada o en la salida* puede ayudar a distinguir diferentes tipos de grupos de datos, pero no se puede confiar en ellos para distinguirlos. Como ejemplo, dos o más conjuntos de atributos de datos que ocurren en la misma entrada o salida que están físicamente separados por razones estéticas o para facilitar la comprensión, pertenecerán al mismo tipo de grupo de datos si cumplen con la regla anterior. NOTA 3. Consulte la sección 3.5 del Manual de Medición para las definiciones, los principios y las reglas para los *movimientos de datos* que mueven grupos de datos, y la sección 3.5.7 (Ejemplos 2, 3, 4 y 5) y 3.5.11 para las excepciones a estas reglas para movimientos de datos, según la regla c anterior
3.5.2	**Entrada (E)** **Principios** a) Una Entrada moverá un solo grupo de datos que describa un solo objeto de interés de un usuario funcional a través del límite y al proceso funcional del cual forma parte la Entrada. Si la entrada a un proceso funcional comprende más de un grupo de datos, cada uno describe un objeto de interés diferente, identifique una Entrada para cada grupo de datos único en la entrada. (Consulte también la sección 3.5.7 sobre "Movimientos de datos únicos".) b) Una entrada no debe salir de los datos a través del límite, o leer o escribir datos desde/hacia el almacenamiento persistente. **Reglas** a) El grupo de datos de una Entrada desencadenante puede consistir en un solo atributo de datos que simplemente informa al software que "ha ocurrido un evento Y". Muy a menudo, especialmente en el software de aplicación empresarial, el grupo de datos de la Entrada desencadenante tiene varios atributos de datos que informan al software que "ha ocurrido un evento ‘Y’ y aquí están los datos sobre ese evento en particular". b) Las señales de reloj que desencadenan eventos siempre serán externas al software que se está midiendo. Por lo tanto, por ejemplo, un evento de marca de reloj que ocurre cada 3 segundos se asociará con una Entrada que mueve un grupo de datos de un atributo de datos. Tenga en cuenta que no hace ninguna diferencia si el evento desencadenante se genera periódicamente por hardware o por otra pieza de software fuera del límite del software que se mide. c) A menos que sea necesario un proceso funcional específico, no se considerará que la obtención de la fecha y/o la hora del reloj del sistema causan una Entrada o cualquier otro movimiento de datos. d) Si una ocurrencia de un evento específico causa la Entrada de un grupo de datos que comprende hasta 'n' atributos de datos de un objeto de interés en particular y el FUR permite que otras ocurrencias del mismo evento puedan causar una Entrada de un grupo de datos que tenga valores para los atributos de solo un subconjunto de los atributos 'n' del objeto de interés, se identificará una Entrada, moviendo un grupo de datos que comprende todos los atributos de datos 'n'.

	e) Al identificar Entradas en una pantalla que permite a los usuarios humanos funcionales ingresar datos en procesos funcionales, analice solo las pantallas que están llenas de datos. Ignore cualquier pantalla que esté formateada, pero, por lo demás, "en blanco", excepto los posibles valores predeterminados, e ignore todos los encabezados de campo y otros que permiten a los usuarios humanos comprender los datos de entrada requeridos. NOTA. Puede ser necesario tener en cuenta los encabezados de campo y otros al medir FUR para los cambios en las Entradas; consulte la sección 4.4.1
3.5.3	**Salida (X)** **Principios** a) Una Salida moverá un solo grupo de datos que describa un único objeto de interés del proceso funcional del cual la Salida forma parte a través de la frontera a un usuario funcional. Si la salida de un proceso funcional comprende más de un grupo de datos, identifique una Salida para cada grupo de datos único en la salida. (Consulte también la sección 3.5.7 sobre "Movimientos de datos únicos".) b) Una Salida no debe ingresar datos a través de la frontera, o leer o escribir datos desde/hacia el almacenamiento persistente. **Reglas** a) Una consulta que genere texto fijo, (donde 'fijo' significa que el mensaje no contiene valores de datos variables, por ejemplo, el resultado de presionar un botón para 'Términos y Condiciones' en un sitio web de compras), se modelará con una Salida para la salida de texto fija. NOTA: Para obtener información sobre la salida de la funcionalidad ‘Ayuda’, consulte la “Guía de Medición para las Aplicaciones de Negocios”. Para la salida de mensajes relacionados con condiciones de error o confirmación de éxito, consulte la sección 3.5.11 de este Manual de medición. b) Si una Salida de un proceso funcional mueve un grupo de datos que comprende hasta 'n' atributos de datos de un objeto de interés en particular y el FUR permite que el proceso funcional tenga una ocurrencia de una Salida que mueva un grupo de datos que tenga valores para atributos de solo un subconjunto de los atributos 'n' del objeto de interés, se identificará una Salida, moviendo un grupo de datos que comprende todos los atributos de datos 'n'. c) Al identificar Salidas, ignore todos los campos y otros encabezados que permitan a los usuarios humanos comprender los datos de salida. NOTA: Es posible que sea necesario tener en cuenta los encabezados de campo y otros al medir FUR para cambios en Salidas; consulte la sección 4.4.1
3.5.4	**Lecturas (R)** **Principios** a) Una Lectura moverá un solo grupo de datos que describa un único objeto de interés desde el almacenamiento persistente a un proceso funcional del que forma parte la Lectura. Si el proceso funcional debe recuperar más de un grupo de datos del almacenamiento persistente, identifique una Lectura para cada grupo de datos único que se recupere. (Consulte también la sección 3.5.7 sobre "Movimientos de datos únicos".) b) Una Lectura no recibirá ni enviará datos a través de su frontera, ni escribirá datos en el almacenamiento persistente. c) Durante un proceso funcional, el movimiento o la manipulación de constantes o variables que son internas al proceso funcional y que pueden ser cambiadas solo por un programador, o el cálculo de resultados intermedios en un cálculo, o de datos almacenados por un proceso funcional que resulta solo de la implementación, en lugar de la FUR, no se considerará como movimientos datos de Lectura.

	d) Un movimiento de datos de Lectura siempre incluye cualquier función de "solicitud de Lectura" (por lo tanto, un movimiento de datos por separado nunca se contabilizará para cualquier funcionalidad de "solicitud de Lectura"). Ver también la sección 3.5.9 **Reglas** a) Identifique una Lectura cuando, según el FUR, el software que se está midiendo debe recuperar un grupo de datos del almacenamiento persistente. b) No identifique una Lectura cuando el FUR del software que se está midiendo especifique a cualquier usuario funcional de software o hardware como la fuente de un grupo de datos, o como el medio de recuperar un grupo de datos almacenado de forma persistente. (Para este caso, vea los principios y reglas para Entradas y Salidas).
3.5.5	**Escrituras (W)** **Principios** a) Una Escritura moverá un solo grupo de datos que describa un único objeto de interés del proceso funcional del cual la Escritura forma parte del almacenamiento persistente. Si el proceso funcional debe mover más de un grupo de datos al almacenamiento persistente, identifique una Escritura para cada grupo de datos único que se mueva al almacenamiento persistente. (Consulte también la sección 3.5.7 sobre "Movimientos de datos únicos".) b) Una Escritura no recibirá o enviará datos a través de la frontera, ni leerá datos del almacenamiento persistente. c) El requisito de eliminar un grupo de datos del almacenamiento persistente se medirá como un solo movimiento de datos de Escritura. d) Lo siguiente no se considerará como un movimiento de datos de Escritura: • El movimiento o manipulación de cualquier dato que no existía al inicio de un proceso funcional y que no se haya hecho persistente cuando el proceso funcional está completo; • Creación o actualización de variables o resultados intermedios que son internos al proceso funcional; • Almacenamiento de datos por un proceso funcional que resulta solo de la implementación, en lugar del FUR. (Un ejemplo sería el almacenamiento de datos temporalmente durante un proceso de clasificación grande en un trabajo procesado por lotes). **Reglas** a) Identifique una Escritura cuando, de acuerdo con el FUR, el software que se está midiendo debe mover un grupo de datos a un almacenamiento persistente. b) No identifique una Escritura cuando el FUR del software que se está midiendo especifique a cualquier usuario funcional de software o hardware como destino del grupo de datos o como medio para almacenar el grupo de datos. (Para este caso, vea los principios y reglas para Entradas y Salidas).
3.5.6	**Manipulación de datos asociada con movimientos de datos.** **Principios** Toda la manipulación de datos en un proceso funcional se asociará con los cuatro tipos de movimiento de datos (E, X, R y W). Por convención, se supone que los movimientos de datos de un proceso funcional también tienen en cuenta la manipulación de los datos del proceso funcional. **Reglas** a) Un movimiento de datos de Entrada representa toda la manipulación de datos para permitir que un usuario funcional ingrese un grupo de datos (por ejemplo, manipulaciones de formato y presentación) y que se validen. b) Un movimiento de datos Salida representa toda la manipulación de datos para crear los atributos de datos de un grupo de datos que se deben generar y/o permitir que el grupo de datos se

	genere (por ejemplo, manipulaciones de presentación y formato) y que se dirijan al usuario funcional deseado. c) Un movimiento de datos de Lectura representa todos los cálculos y/o procesamientos lógicos necesarios para recuperar un grupo de datos del almacenamiento persistente. d) Un movimiento de datos de Escritura representa todos los cálculos y/o procesamientos lógicos para crear o actualizar un grupo de datos a ser escrito, o para eliminar un grupo de datos. e) La manipulación de datos asociada con cualquiera de estos movimientos de datos no incluye ninguna manipulación de datos que sea necesaria después de que el movimiento de datos se haya completado con éxito, ni incluye ninguna manipulación de datos asociada con ningún otro movimiento de datos.
3.5.7	**Movimientos de datos únicos y posibles excepciones** **Reglas** Nótese bien que todas las reglas COSMIC 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. Esta convención se sigue en las reglas a), b) y c) a continuación, pero en la regla d) incluimos "tipo" donde es útil distinguir un "tipo" de una "ocurrencia". a) A menos que los Requisitos de Usuario Funcional sean los que se indican en las reglas b) o c), todos los datos que describan cualquier objeto de interés que deba ingresarse en un proceso funcional se identificarán como un grupo de datos movido por una Entrada. NOTA: un proceso funcional puede, por supuesto, tener múltiples Entradas, cada una de ellas con datos en movimiento que describen un objeto de interés diferente. La misma regla equivalente se aplica a cualquier movimiento de datos de Lectura, Escritura o salida en cualquier proceso funcional. b) Si los Requisitos Funcionales del Usuario especifican que se deben ingresar diferentes grupos de datos en un proceso funcional, uno por cada usuario funcional diferente, donde cada grupo de datos describe el mismo objeto de interés, se identificará una Entrada para cada uno de estos grupos de datos diferentes. La misma regla equivalente se aplica a las Salidas de datos a diferentes usuarios funcionales de cualquier proceso funcional. NOTA: Cualquier proceso funcional tendrá una sola Entrada desencadenante. c) Si los Requisitos Funcionales del Usuario especifican que se deben mover diferentes grupos de datos del almacenamiento persistente a un proceso funcional, cada uno describiendo el mismo objeto de interés, se identificará una Lectura para cada uno de estos grupos de datos diferentes. La misma regla equivalente se aplica para Escrituras en cualquier proceso funcional dado. NOTA: Esta regla es análoga a la regla b). En el caso de los FUR para leer diferentes grupos de datos que describen el mismo objeto de interés, probablemente se hayan originado a partir de diferentes usuarios funcionales. En el caso de que los FUR escriban diferentes grupos de datos, es probable que estén disponibles para que los lean diferentes usuarios funcionales. d) Las ocurrencias repetidas de cualquier tipo de movimiento de datos cuando se está ejecutando no se contabilizarán. Esto se aplica incluso si varias ocurrencias del tipo de movimiento de datos difieren en su ejecución debido a que diferentes valores de los atributos de datos del grupo de datos movidos dan como resultado diferentes caminos de procesamiento que se siguen a través del tipo de proceso funcional.
3.5.9	**Un proceso funcional que requiere datos de un usuario funcional** **Reglas**

	a) Un proceso funcional debe obtener un grupo de datos a través de un movimiento de datos de Entrada de un usuario funcional, cuando el proceso funcional no necesita decirle al usuario funcional qué datos 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 el proceso funcional; • cuando un proceso funcional, después de haber recibido un grupo de datos a través de una Entrada desencadenante, espera, expectante la llegada de un grupo de datos adicional del usuario funcional a través de una Entrada (puede ocurrir cuando un usuario funcional humano ingresa datos al software de aplicación de negocios); • cuando un proceso funcional, después de haber comenzado, solicita al usuario funcional, "envíeme sus datos ahora, si tiene alguno" y el usuario funcional envía sus datos; • cuando un proceso funcional, después de haber comenzado, inspecciona el estado de un usuario funcional y recupera los datos que requiere. En los últimos dos casos (que generalmente ocurren en el software de "sondeo" en tiempo real), por convención, no se identificará ninguna Salida del proceso funcional para obtener los datos requeridos. El proceso funcional simplemente necesita enviar un mensaje de solicitud a un usuario funcional y la funcionalidad de ese mensaje de solicitud se considera parte de la Entrada. El proceso funcional sabe qué datos esperar. Solo se identificará una Entrada para este caso. b) Cuando un proceso funcional necesita obtener los servicios de un usuario funcional (por ejemplo, para obtener datos) y *el usuario funcional necesita que se le diga qué enviar* (generalmente donde el usuario funcional es otra pieza de software fuera del alcance del software que se está midiendo), se identificará una Salida seguida de un movimiento de datos de Entrada. La Salida envía la solicitud de los datos específicos; la Entrada recibe los datos devueltos
3.5.10	**Controlar comandos en aplicaciones con una interfaz humana** **Reglas** En una aplicación con una interfaz humana, los "comandos de control" se ignorarán, ya que no implican ningún movimiento de datos sobre un objeto de interés.
3.5.11	**Mensajes de error / confirmación y otras indicaciones de condiciones de error** **Reglas** a) Se identificará una Salida para tener en cuenta todos los tipos de mensajes de error/confirmación emitidos por cualquier proceso funcional del software que se esté midiendo de todas las causas posibles de acuerdo con su FUR, p. ej. éxitos o fracasos de: validación de los datos ingresados o para una llamada para recuperar datos o hacer que los datos sean persistentes, o para la respuesta de un servicio solicitado de otra pieza de software. NOTA: Si el FUR del proceso funcional no requiere que se emita ningún tipo de mensaje de error/confirmación, no identifique ninguna Salida correspondiente. b) Si un mensaje a un usuario funcional humano proporciona datos además de confirmar que los datos ingresados han sido aceptados o que los datos ingresados son erróneos, entonces estos datos adicionales deben identificarse como un grupo de datos movido por una Salida de la manera normal, además de la Salida del error/confirmación. c) Todos los demás datos, emitidos o recibidos por el software que se está midiendo, a/desde sus usuarios funcionales de hardware o software deben analizarse de acuerdo con el FUR como Salidas o Entradas respectivamente, de acuerdo con las reglas COSMIC normales, independientemente de si los valores de datos indican una condición de error. d) Las Lecturas y Escrituras se consideran para tener en cuenta, cualquier informe asociado de condiciones de error. Por lo tanto, no se identificará ninguna Entrada al proceso funcional que se esté midiendo para ninguna indicación de error recibida como resultado de una Lectura o escritura de datos persistentes.

	e) No se identificará ninguna Entrada o Salida para ningún mensaje que indique una condición de error que podría emitirse mientras se utiliza el software que se está midiendo, pero que no está obligado a ser procesado de ninguna manera por el FUR de ese software, p. ej. Un mensaje de error emitido por el sistema operativo.
4.3.1	**Agregando los resultados de medición** **Reglas** a) Para cualquier proceso funcional, el tamaño funcional de los movimientos de datos individuales se agregará en un solo valor de tamaño funcional en unidades de CFP al sumarlos. Tamaño (procesos funcionales) = Σ tamaño(Entradas) + Σ tamaño(Salidas) + Σ tamaño(Lecturas) + Σ tamaño(Escrituras) b) Para cualquier proceso funcional, el tamaño funcional de los cambios en sus Requisitos funcionales del usuario se agregará a partir del tamaño de los movimientos de datos que se han agregado, modificado o eliminado en el proceso funcional para dar un tamaño del cambio en unidades CFP, de acuerdo con a la siguiente fórmula. Tamaño (Cambio(procesos funcionales)) = Σ tamaño (mov. de datos agregado) + Σ tamaño (mov. de datos modificados) + Σ tamaño (mov. de datos eliminados.) Para más información sobre agregar tamaño funcional, consulte la sección 4.3.2. Para medir el tamaño del software modificado, consulte la sección 4.4.2 c) El tamaño de una pieza de software dentro de un alcance definido se obtendrá agregando los tamaños de los procesos funcionales para la pieza, sujeto a las reglas e) y f) a continuación. d) El tamaño de cualquier cambio en una pieza de software dentro de un alcance definido se obtendrá agregando los tamaños de todos los cambios a todos los procesos funcionales para la pieza, sujeto a las reglas e) y f) a continuación e) Los tamaños de las piezas de software o de los cambios en las piezas de software solo se pueden sumar si se miden al mismo nivel de granularidad de proceso funcional de su FUR. f) Los tamaños de las piezas de software y/o los cambios en los tamaños de las piezas de software dentro de cualquier capa o de diferentes capas se agregarán juntas solo si tiene sentido hacerlo, para el propósito de la medición. g) El tamaño de una pieza de software se obtiene sumando los tamaños de sus componentes (independientemente de cómo se descomponga la pieza) y eliminando las contribuciones de tamaño de los movimientos de datos entre componentes. h) Solo se identificará una Salida para todos los mensajes de error / confirmación emitidos por cualquier proceso funcional a un usuario funcional humano. i) 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 a través de la extensión local se informará por separado como se describe en la sección 5.1 y NO se agregará al tamaño obtenido por el método estándar, medido en CFP (ver más adelante en la sección 4.5).
4.4.1	**Modificar un movimiento de datos** **Reglas** a) Se considera que un movimiento de datos se modifica funcionalmente si se aplica al menos uno de los siguientes: • el grupo de datos movido se modifica, • se modifica la manipulación de datos asociada. b) Un grupo de datos se modifica si se aplica al menos uno de los siguientes:

	• uno o más atributos nuevos se agregan al grupo de datos, • uno o más atributos existentes se eliminan del grupo de datos, • se modifican uno o más atributos existentes, p. ej. en significado o formato (pero no en sus valores) c) Una manipulación de datos se modifica si se cambia funcionalmente de alguna manera
4.4.2	**Tamaño del software funcionalmente modificado** **Regla** Después de cambiar funcionalmente una pieza de software: Nuevo tamaño total (pieza de software modificada) = Tamaño total anterior (pieza de software) + Σ tamaño (movimientos de datos agregados) - Σ tamaño (movimientos de datos eliminados)
5.1	**Etiquetado de medidas COSMIC** **Reglas** El resultado de una medición COSMIC se anotará como 'x CFP (v)', donde: • "x" representa el valor numérico del tamaño funcional, • "v" representa la identificación de la versión del método COSMIC estándar utilizado para obtener el valor numérico de tamaño funcional "x" NOTA: Si se usó un método de aproximación local para obtener la medición, pero de lo contrario la medición se realizó utilizando las convenciones de una versión COSMIC estándar, se utilizará la convención de etiquetado anterior, pero el uso del método de aproximación se debe anotar en otra parte - vea la sección 5.2.
5.1	**Etiquetado de extensiones locales COSMIC** **Regla** El resultado de una medición COSMIC usando extensiones locales se anotará como: "**x** CFP (v.) + Z Local FP", donde: • "**x**" representa el valor numérico obtenido al agregar todos los resultados de medición individuales de acuerdo con el método COSMIC estándar, versión v, • "v" representa la identificación de la versión del método COSMIC estándar utilizado para obtener el valor numérico de tamaño funcional "x" "z" representa el valor numérico obtenido al agregar todos los resultados de medición individuales obtenidos de extensiones locales al método COSMIC
5.2	**Informe de mediciones COSMIC** **Regla** Además de las mediciones reales, registradas como en 5.1, se deben registrar algunos o todos los siguientes atributos de cada medición, dependiendo del propósito de la medición y el nivel deseado de comparabilidad con otras mediciones, p. para fines de evaluación comparativa. a) Identificación del componente de software medido (nombre, ID de versión o ID de configuración). b) Las fuentes de información utilizadas para identificar el FUR utilizado para la medición. c) El dominio del software. d) Una descripción de la arquitectura de las capas en las que se realiza la medición, si corresponde. e) Una declaración del propósito de la medición.

	f) Una descripción del alcance de la medición, y su relación con el alcance general de un conjunto relacionado de mediciones, si corresponde. (Utilice las categorías de alcance genérico en la sección 2.2). g) El patrón de medición (estrategia) utilizado (COSMIC o local), con el modo de procesamiento (en línea 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 realizó la medición (especialmente si la medición es una estimación basada en requisitos incompletos, o se realizó sobre la base de la funcionalidad realmente entregada). k) El margen de error objetivo o creído de la medición. l) Indica si se usó el método de medición COSMIC estándar, y/o una aproximación local al método estándar, y/o si se usaron extensiones locales (ver sección 4.5). Utilice las convenciones de etiquetado de las secciones 5.1 o 5.2. m) Una indicación de si la medición es de funcionalidad desarrollada o entregada (la funcionalidad 'desarrollada' se obtiene mediante la creación de un nuevo software; la funcionalidad 'entregada' incluye la funcionalidad 'desarrollada' y también incluye la funcionalidad obtenida por otros medios que no sean la creación de un nuevo software, es decir, incluyendo todas las formas de reutilización de software existente, implementación de paquetes de software, uso de parámetros existentes para agregar o cambiar funcionalidades, etc. n) Una indicación de si la medición es de una funcionalidad recientemente proporcionada o es el resultado de una actividad de "mejora" (es decir, la suma es de funcionalidad agregada, modificada y eliminada - ver 4.4). o) El número de componentes principales, si corresponde, cuyos tamaños se han sumado para el tamaño total registrado. p) El porcentaje de funcionalidad implementada por el software reutilizado. q) Para cada alcance dentro del alcance de medición general, una matriz de medición, como se especifica en el Apéndice A. r) El nombre del Medidor y cualquier calificación de certificación COSMIC; La fecha de la medición.

APENDICE E - PRINCIPALES CAMBIOS DE LA VERSIÓN 4.0 A LAS VERSIONES V4.0.1 Y A V4.0.2

Este Apéndice contiene un resumen de los principales cambios realizados en la evolución del método de medición del tamaño funcional COSMIC desde la versión 4.0 a la v4.0.1 y hasta la presente v4.0.2.

Para rastrear la evolución anterior del método, consulte el Manual de medición para cada versión anterior del método (2.2, 3.0 y 3.0.1 y 4.0).

Un "MUB" es un Boletín de actualización de métodos, publicado entre las principales versiones del Manual de medición para anunciar y explicar los cambios propuestos.

E1: Cambios principales de V4.0 a v4.0.1

V4.0.1 Ref	Cambio
Prefacio	El documento de Introducción debe leerse primero si es un principiante o está convirtiendo, esto se ha enfatizado al agregar un título al párrafo existente
1.2.3	La definición de un "requisito no funcional" se ha aclarado y ahora no incluye requisitos y restricciones del proyecto. La figura 1.3 ha sido modificada correspondientemente.
1.3.3	Se ha agregado una nueva sección 1.3.3 "Tipos versus ocurrencias". COSMIC nunca ha definido el "tipo" y la "ocurrencia" explícitamente y parece que los Medidores a veces todavía tienen dificultades para comprender la diferencia,
3.2.1	La definición de "proceso funcional" fue ambigua en un momento dado que no estaba claro si "único" se refiere al "conjunto de movimientos de datos" o "a la parte elemental de la FUR". Se ha resuelto agregando una coma después de "movimientos" y cambiando dos ocurrencias de "eso" a "estos".
3.3.1	La definición de un "grupo de datos" se ha simplificado ya que era innecesariamente compleja. Dos de los principios han sido eliminados ya que no agregaron valor.
3.3.1	Se ha revisado la definición de un "objeto de interés" y se ha agregado una nota, ambas con el objetivo de aclarar la definición
3.5.1	Las definiciones de un "movimiento de datos", "Entrada", "Salida", "Lectura" y "Escritura" ahora establecen que se considera que cada uno "tiene en cuenta" (no "incluye") ninguna manipulación de datos asociada

V4.0.1 Ref	Cambio
3.5.6	El texto ahora enfatiza que las reglas en esta sección son relevantes solo para la medición de cambios a FUR existentes. Las reglas que definen qué tipos de manipulaciones de datos están asociadas con qué movimientos de datos ahora dejan en claro que la manipulación de datos está "asociada con", no "incluida" en los movimientos de datos.
3.5.7	Las reglas para la "singularidad del movimiento de datos y posibles excepciones" se han modificado ya que eran complejas. Se ha corregido un error (dos movimientos de datos no pueden considerarse diferentes debido a que tienen diferentes FUR para su manipulación de datos asociada, ya que esto contradiría el principio de que 'toda manipulación de datos en un

	proceso funcional se asociará con los cuatro tipos de movimiento de datos (E, X, R y W) '). Se agregaron dos ejemplos y los ejemplos se volvieron a secuenciar para alinearse con las reglas revisadas. También se han hecho algunas simplificaciones editoriales.
3.5.11	Se ha aclarado la definición de "mensaje de error/confirmación". Parte de la definición anterior se ha convertido en una regla.
4.3.1	La regla g) sobre cómo obtener el tamaño de una pieza de software a partir de los tamaños de sus componentes se ha cambiado de una declaración "doble negativa" a una declaración positiva. La regla se ha extendido sobre cómo agregar mensajes de error/confirmación.
Varios	Las referencias a las normas ISO/IEC 14143/1 sobre la Definición de conceptos (de métodos FSM) y a ISO/IEC 19761 que define los conceptos principales del método COSMIC fueron confusas y ahora se han corregido.
Glosario	Las definiciones de "entrada" y "salida" se han simplificado ya que eran innecesariamente complejas.

E2: Principales cambios de v4.0.1 a v4.0.2

Nota. La naturaleza de un cambio se indica mediante:

- "Método" cuando una definición o regla del método COSMIC se ha agregado o se ha cambiado para mejorar la comprensión. (Los cambios o adiciones a las Notas se consideran cambios "editoriales").
- "Editorial" cuando se ha modificado algún texto para mejorar la comprensión. También se han realizado muchas mejoras editoriales menores además de las que se enumeran a continuación.
- "Corrección" cuando se ha corregido un error en la versión anterior v4.0.1 de esta Guía.

V4.0.2 Ref	Naturaleza del cambio	Cambio
-	Editorial	El nombre provisional ‘Guía para la medición aproximada del tamaño funcional COSMIC’ se ha cambiado a su nombre definitivo “Guía para la medición temprana o rápida por un enfoque de aproximación del tamaño funcional COSMIC”.
Prefacio		Se ha agregado un resumen de los principales cambios para v4.0.2.
1.2	Editorial	En la versión 4.0 del método, restringimos cómo usaríamos el término "Requisitos funcionales del usuario" (FUR) en el método COSMIC. La restricción se ha agregado ahora como una Nota 2 a la definición del término.
1.2, 1.3	Correcciones	En varios lugares del Capítulo 1, especialmente en dos Principios del Modelo de Contexto del Software, el término "FUR" ha sido reemplazado por "requisitos funcionales" para reflejar el uso ahora restringido de "FUR". Estos cambios deberían haberse realizado cuando se introdujo la interpretación restringida en v4.0.
1.3.2	Método Editorial	Se ha agregado un Principio que dice "El tamaño de un proceso funcional es igual al recuento total de sus movimientos de datos". Esto solo se ha establecido como norma en la sección 4.3.1. El Principio relativo al rango de tamaño de un proceso funcional ahora establece explícitamente que no hay un límite superior para su tamaño. Se ha agregado una nota al Modelo Genérico de Software que explica que el modelo no tiene la intención de describir la secuencia física de pasos mediante los cuales se ejecuta un proceso funcional.
1.3.3	Editorial	Esta sección se ha reestructurado y la explicación del término "tipo" se ha mejorado para ayudar a comprender, en particular para ayudar a identificar cuándo los usuarios funcionales son del mismo tipo. La explicación del ejemplo 2 en tiempo real también se ha mejorado.
2.0	Editorial	El Resumen del Capítulo se ha mejorado para que sea más simple y se alinee mejor con el contenido del capítulo.
2.2.1	Editorial	El texto del ejemplo de negocios ilustrado en la figura 2.1 se ha mejorado para mayor claridad.
2.2.2	Editorial	La referencia www obsoleta a la arquitectura AUTOSAR ha sido reemplazada.
2.2.3	Editorial	En la Nota 2 de la definición de "Niveles de descomposición", la frase "solo puede ser" se ha cambiado a "solo", ya que el uso de "puede ser" no es correcto en este contexto. Se ha agregado una Nota 3 de que los niveles de descomposición pueden corresponder a las capas de la arquitectura del software, pero no necesariamente.
2.3	Editorial	El título de la sección se ha cambiado porque no es necesario identificar el "Almacenamiento persistente". La parte inicial de 2.3 y la sección 2.3.1 se han reestructurado en una secuencia más lógica.
2.3.1	Método	La definición de "usuario funcional" se ha hecho más específica (MUB 12).

2.3.1	Método	Se ha agregado una Nota a las reglas de un "usuario funcional" para ayudar a comprender. El ejemplo 1 presentado en v4.0.1 no fue realmente un ejemplo. Se ha reescrito como texto y los siguientes ejemplos se han vuelto a numerar. Se han agregado más explicaciones de "usuario funcional". El ejemplo en tiempo real 2 se ha "modernizado" al cambiar el tema de un teléfono móvil con teclado (ahora menos común) a una copiadora. El ejemplo en tiempo real 4 se ha "modernizado" para que la salida sea a una pantalla de tablero, en lugar de a cuatro LED.
2.3.2	Método	Se ha agregado una explicación adicional del almacenamiento persistente y su relación con los archivos o las bases de datos.
2.4.1	Método	La definición de "Nivel de granularidad" se ha ampliado ligeramente para dejar en claro que se aplica a la descripción de "cualquier parte de" una pieza de software, por lo que no necesariamente a la descripción completa.
2.4.3	Corrección	La definición de "Nivel de granularidad del proceso funcional" se ha mejorado para facilitar la comprensión, incluida la corrección por el uso ahora restringido del término "FUR". Las reglas para 'Nivel de granularidad del proceso funcional' se han renombrado como 'Niveles de granularidad para medir un proceso funcional', para reflejar los hechos de que medir un proceso funcional requiere FUR en dos niveles de granularidad, es decir, el del proceso funcional y de sus movimientos de datos, y esa regla b) también se refiere a la medición aproximada a niveles más altos de granularidad.
2.4.3	Editorial	Se deja en claro que todo el ejemplo del "Sistema de pedidos Everest" se refiere al nivel de granularidad del FUR (no al nivel de descomposición del software). El propósito de los ejemplos también se ha aclarado.
3.2.1	Método	La definición de "Evento desencadenante" se ha modificado para dejar en claro que solo el primer grupo de datos generado por cualquier usuario funcional será movido por una Entrada desencadenante. (MUB 14).
3.2.1	Método	En el segundo punto de la definición de un Proceso funcional, "puede" ha sido reemplazado por "debe". "Puede" es técnicamente correcto porque, como se explica en el Prólogo, en la terminología ISO "puede" significa "está permitido". Sin embargo, el uso de "debe" aclara que esta es una parte obligatoria de la definición. Se ha agregado una Nota 4 a la definición para explicar el uso de "único" en la cláusula a) de la definición.
3.2.1	Editorial	En la definición de "Entrada desencadenante", se ha agregado una Nota para explicar que la frase "que el proceso funcional necesita comenzar a procesar" es el resultado del GSM lógico y no necesariamente significa que el procesamiento comienza físicamente cuando se ingresan los datos. En la definición de "Entrada desencadenante" en el Glosario, se ha eliminado una Nota. Esta nota realmente pertenece a la definición de un proceso funcional, donde se indica correctamente
3.2.1	Editorial	En la Figura 3.3, la flecha del grupo de Datos ha sido reemplazada por un símbolo de paralelogramo para darle la misma importancia que los otros símbolos en la Figura.

3.2.2	Editorial	La regla b) para un proceso funcional ha sido eliminada, ya que es superflua. La redacción de (la nueva) regla b) se ha mejorado para facilitar la comprensión. La redacción de (la nueva) regla c) se ha modificado ligeramente para alinearla con la definición de un proceso funcional.
3.2.3	Editorial	La descripción de la medición de procesos funcionales procesados por lotes se ha ampliado para reconocer que los datos de entrada pueden llegar como un flujo de datos transmitido (así como a través de un conjunto de datos almacenado temporalmente). La Figura 3.4 muestra el cambio también.
3.3.1	Método	En la definición de objeto de interés, la frase "procesar y/o mover datos" ha sido reemplazada por "mover un grupo de datos dentro o fuera del software, o hacia o desde el almacenamiento persistente". El cambio tiene como objetivo evitar una posible interpretación de que los objetos de interés pueden identificarse únicamente debido al "procesamiento", es decir, la manipulación de datos. Como consecuencia, la Nota 1 relativa al "procesamiento" ya no es necesaria y se ha eliminado. Se ha agregado una Nota 2 que cuando un usuario funcional envía datos sobre sí mismo, el usuario funcional también es el objeto de interés del grupo de datos enviado. Se ha agregado una Nota 3 de que no hay nada absoluto sobre si una "cosa" es un "objeto de interés" o no (MUB 12).
3.3.1	Editorial	Se ha agregado una Nota a la definición de un "grupo de datos", aclarando que no es necesario que consista en todos los atributos de datos de un objeto de interés (MUB 13).
3.3.1	Editorial	La definición de "atributo de datos" se ha movido de 3.4.1 a 3.3.1, donde pertenece lógicamente.
3.3.2	Método	Se ha agregado una nueva regla: "Identificar diferentes grupos de datos (y, por lo tanto, diferentes objetos de interés) movidos en el mismo proceso funcional". La nueva regla apunta en particular a ayudar a identificar grupos de datos movidos en resultados complejos de procesos funcionales de aplicaciones de negocios. Sin embargo, la regla es válida para el software de todos los dominios y para la entrada y datos movidos hacia o desde el almacenamiento persistente.
3.3.2	Editorial	Casi la totalidad de esta sección ha sido reescrita y se han agregado más ejemplos para aclarar el texto.
3.3.4	Corrección	El texto ha sido revisado para dejar en claro que un usuario funcional puede ser un objeto de interés en el dominio de la aplicación de negocios y en el dominio en tiempo real. Se ha agregado un ejemplo de negocios (en línea con MUB 12).
3.4.1	Editorial	Esta sección ha sido renombrada "Ejemplos de atributos de datos" y los ejemplos han mejorado mucho
3.5.2	Método	La regla c) para una Entrada se ha ampliado para dejar en claro que no es necesaria una Entrada para obtener la fecha actual y/o la hora del sistema operativo. En la regla d) para una Entrada. La palabra "desencadenantes" ha sido reemplazada por "causas", para evitar una interpretación errónea de que esta regla se aplica solo a una Entrada desencadenante

3.5.3	Editorial	La regla b) para una Salida se ha modificado ligeramente para facilitar la comprensión.
3.5.6	Corrección	La definición de manipulación de datos "cualquier cosa que le suceda a los datos ... aparte del movimiento ..." se ha modificado para dejar en claro que esto solo se aplica al procesamiento por (o "dentro") de un proceso funcional.
3.5.7	Editorial	La declaración en el párrafo inicial de que el "Modelo Genérico de Software supone que normalmente ..." está equivocado y se ha eliminado. Podría implicar que este modelo permite excepciones, lo cual no es cierto. El párrafo ahora comienza "Lo más frecuente ..." Se han realizado algunos cambios en la redacción de las reglas de "Movimientos de Datos Únicos" y se ha agregado una Nota para mejorar la comprensión. En particular, la última oración de la cláusula d) se ha eliminado para evitar posibles confusiones. La oración no es realmente incorrecta, pero puede parecer que está en conflicto con las nuevas reglas para distinguir grupos de datos y objetos de interés (ver sección 3.3.2). El ejemplo 10 ha sido cambiado para mayor claridad; es relevante solo para la regla d)
3.5.10	Editorial	Se han eliminado ejemplos de la definición de "Comandos de control". Los ejemplos han sido racionalizados y reescritos para mayor claridad.
3.5.8	Editorial	Se ha agregado un párrafo para explicar que el Medidor primero debe decidir si la recuperación de datos debe realizarse desde el almacenamiento persistente dentro de los límites del software que se está midiendo o desde el almacenamiento persistente accesible a través de otro software.
3.5.11	Editorial	El Ejemplo de Negocios 3 ha sido reescrito para mejorar la facilidad de comprensión.
4.3.1	Método	Las dos partes de la regla g) para la agregar los resultados de medición se han dividido en dos reglas, ya que las dos partes no están relacionadas.
4.4.2	Método	La ecuación para medir el tamaño del software funcionalmente cambiado se ha cambiado de texto a Regla, y se ha agregado un Ejemplo.
4.5.4	Editorial	Se ha agregado una nota para describir que la medición de tamaño funcional COSMIC se ha aplicado con éxito a las mediciones de historias de usuario y sprints en desarrollos ágiles.
Ape. C	Corrección	Ejemplo 6) no se limita al "mismo software", esta frase se ha eliminado.
Ape. F	Editorial	(diciembre de 2017) La indicación de origen se ha corregido durante 1 término para mostrar que es específica de ISO (negrita, sin cursiva) y de 10 términos para mostrar que son específicos de COSMIC (negrita, cursiva). El término "entorno operativo (software)" no se utiliza, por lo que se ha eliminado del Glosario.

APENDICE F – GLOSARIO DE TERMINOS

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

Para muchos de los términos enumerados en el glosario, cuando corresponde, 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 nos preocuparemos por los "tipos" y no por las "ocurrencias". En consecuencia, en los textos eliminaremos el sufijo "tipo" de estos términos en aras de la legibilidad, excepto cuando específicamente necesitemos distinguir el tipo y la ocurrencia. Esta es también la convención adoptada en la definición de la Norma Internacional (ISO / IEC 19761: 2011) del método COSMIC. Ocasionalmente, esta convención genera dificultades al redactar estas definiciones; consulte, por ejemplo, la Nota 3 de la definición de "tipo de movimiento de datos" a continuación, que no aparece en la Norma Internacional.

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

NOTA: Los términos que se usan solo en las "Guías" COSMIC específicas se definen en esas guías; No se muestran a continuación.

En el siguiente:

- los términos utilizados en las definiciones que se definen en otra parte de este glosario están subrayados, para facilitar la referencia cruzada.
- los términos que se originan en la Norma ISO para el método COSMIC (ISO / IEC 19761) o que son específicos del método COSMIC se muestran en ***negrita y cursiva***.
- otros términos que se han adoptado de ISO pero que no son específicos del método COSMIC se muestran en **negrita**.

Aplicación. Un sistema de software para recopilar, guardar, procesar y presentar datos por medio de una computadora.

NOTA: Esta es una adaptación de la definición dada en ISO/IEC 24570: 2005 Ingeniería de software - Método de medición de tamaño funcional NESMA versión 2.1.

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

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

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

Componente funcional base (BFC). Una unidad elemental de los requisitos funcionales del usuario definidos por un método FSM para fines de medición [17].

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

Tipo de componente funcional base (tipo BFC). Una categoría definida de BFC [17]. El método COSMIC tiene cuatro tipos de BFC, la Entrada, Salida, Lectura y Escritura (tipos).

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

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

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

Comando de control. Un comando que permite a los usuarios funcionales humanos controlar su uso del software pero que no implica ningún movimiento de datos sobre un objeto de interés del 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.

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

NOTA: La unidad de medida se conocía como "Cfsu" (unidad de tamaño funcional COSMIC) antes de la v3.0 del método.

Tipo de atributo de datos (sinónimo "tipo de elemento de datos"). El paquete de información más pequeño, dentro de un tipo de grupo de datos identificado, que tiene un significado desde la perspectiva de los Requisitos del Usuario Funcional del software.

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

NOTA: "Grupo de datos" no significa necesariamente "el conjunto de *todos* los atributos de datos que describen un solo objeto de interés". El FUR de una pieza de software puede especificar grupos de datos que se formarán a partir de cualquier combinación de atributos de datos que todos describan el mismo objeto de interés, según lo necesiten los diferentes procesos funcionales.

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

Tipo de movimiento de datos. Un componente funcional base que mueve un solo tipo de grupo de datos.

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

NOTA 2: Para fines de medición, se considera que cada movimiento de datos tiene en cuenta cierta manipulación de datos asociada; consulte el Manual de medición para obtener 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 también se aplica a las definiciones de Entrada, Salida, Lectura y Escritura.

E. Abreviatura de "Tipo de Entrada".

Tipo de 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 requiere.

NOTA: Se considera que un tipo de Entrada representa cierta manipulación de datos asociada; consulte el Manual de medición para obtener más detalles.

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

NOTA: Cualquier Salida que pueda incluir indicaciones de falla, pero que no esté destinada a un usuario funcional humano, no es un mensaje de error/confirmación.

Tipo de evento. Algo que sucede.

Tipo de Salida. Un movimiento de datos que mueve un grupo de datos desde un proceso funcional a través de la frontera hasta el usuario funcional que lo requiere.

NOTA: Se considera que un tipo de Salida da cuenta de cierta manipulación de datos asociada; consulte el Manual de medición para obtener más detalles.

Tipo de proceso funcional

a) Un conjunto de movimientos de datos, que representan una parte elemental de los Requisitos del Usuario Funcional para el software que se está midiendo, que es único dentro de estos FUR y que se puede definir independientemente de cualquier otro proceso funcional en estos FUR

b) Un proceso funcional solo tendrá una entrada desencadenante. Cada proceso funcional comienza a procesarse al recibir un grupo de datos movido por el movimiento de datos de Entrada desencadenante del proceso funcional.

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

NOTA 1: Cuando se implementa, es una *ocurrencia* de un proceso funcional que comienza a *ejecutarse* al recibir 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 adicionales además de la Entrada desencadenante.

NOTA 3: Si un usuario funcional envía un grupo de datos con errores, p. ej. debido a que un usuario sensor funciona mal o los datos ingresados por un humano tienen errores, generalmente es tarea del proceso funcional determinar si el evento realmente ocurrió y/o si los datos ingresados son realmente válidos y cómo responder.

NOTA 4: Un proceso funcional es “único” (como en a) anterior), y su tamaño total debe incluirse en el tamaño del FUR, si es iniciado por una Entrada desencadenante que resulta originalmente de un evento desencadenante que se distingue como único dentro de la FUR. Dos o más procesos funcionales dentro del mismo FUR pueden ser únicos, aunque compartan alguna funcionalidad común. Consulte la sección 3.2.7 para ver ejemplos de procesos funcionales con funcionalidad compartida.

Nivel funcional de granularidad del proceso. Un nivel de granularidad de la descripción de una pieza de software en el que

- el usuario funcional (-tipos) son humanos individuales o dispositivos de ingeniería o piezas de software (y no ningún grupo de estos) Y
- se producen eventos únicos (-tipos) a los que la pieza de software debe responder (y no cualquier nivel de granularidad en el que se definen grupos de eventos). Ver Nota 3 a continuación.

NOTA 1: En la práctica, la documentación del software a menudo describe los requisitos funcionales de diferentes partes del software a diferentes niveles de granularidad, especialmente cuando la documentación aún está evolucionando. Los procesos funcionales pueden revelarse en cualquiera de estos niveles de granularidad.

NOTA 2: "Los 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: Los 'grupos de eventos' podrían, por ejemplo, indicarse en una declaración de requisitos funcionales con un alto nivel de granularidad mediante un flujo de entrada a un sistema de software de contabilidad etiquetado como 'transacciones de ventas' o por un flujo de entrada a un software de aviónica sistema etiquetado como 'comandos piloto'.

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

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

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

Usuario Funcional. Un (tipo de) usuario que se identifica en los Requisitos funcionales del usuario de una pieza de software que se mide como remitente y / o destinatario de los datos procesados por el software.

Requisitos funcionales del usuario (FUR). Un subconjunto de los requisitos del usuario. Requisitos que describen lo que debe hacer el software, en términos de tareas y servicios.

NOTA 1: Los Requisitos del Usuario Funcional se relacionan con, pero no se limitan a:

- transferencia de datos (por ejemplo, datos del cliente de entrada, señal de control de envío);
- transformación de datos (por ejemplo, Calcular interés bancario, Derivar temperatura promedio);
- almacenamiento de datos (por ejemplo, almacenar pedido del cliente, registrar la temperatura ambiente a lo largo del tiempo);
- recuperación de datos (por ejemplo, Lista de empleados actuales, Recuperar posición de aeronave).

Los ejemplos de requisitos del usuario que no son Requisitos del Usuario Funcional incluyen, entre otros (aunque algunos de estos pueden convertirse en FUR verdaderos para cuando la solución esté completamente definida):

- restricciones de calidad (por ejemplo, usabilidad, confiabilidad, eficiencia y portabilidad);
- restricciones organizacionales (por ejemplo, ubicaciones para operación, hardware objetivo y cumplimiento de estándares);
- limitaciones medioambientales (por ejemplo, interoperabilidad, seguridad, privacidad y seguridad);
- restricciones de implementación (por ejemplo, lenguaje de desarrollo, cronograma de entrega).

NOTA 2: en los documentos COSMIC, el término "FUR" está restringido a los requisitos funcionales del usuario que:

- se derivan de los artefactos de software disponibles (requisitos, diseños, artefactos físicos, etc.);
- se ajustan, si es necesario, por supuestos para superar las incertidumbres en los artefactos disponibles;
- contener toda la información necesaria para una medición de tamaño funcional COSMIC.

  De lo contrario, usamos expresiones como requisitos funcionales "," requisitos reales "o" artefactos físicos ", etc., según el contexto

entrada. Los datos movidos por todas las Entradas de un proceso funcional dado.

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

Nivel de descomposición. Cualquier nivel resultante de dividir una pieza de software en componentes (llamado 'Nivel 1', por ejemplo), luego de dividir componentes en subcomponentes ('Nivel 2'), luego de dividir subcomponentes en sub-subcomponentes (Nivel 3 '), etc.

NOTA 1: No debe confundirse con el "nivel de granularidad" que se refiere al nivel de detalle de los requisitos.

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

NOTA 3: Los diferentes niveles de descomposición de una pieza de software pueden corresponder a diferentes "vistas" de las capas del software, p. ej. como en la figura 2.4. Sin embargo, el software puede descomponerse en "niveles" independientemente de si está diseñado o no con un modelo de arquitectura en capas.

Nivel de granularidad. Cualquier nivel de expansión de la descripción de cualquier parte 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 mayor nivel de expansión, la descripción de la funcionalidad de la pieza de software tiene un nivel de detalle mayor y uniforme.

NOTA: Los medidores deben tener en cuenta que cuando los requisitos evolucionan temprano en la vida de un proyecto de software, en cualquier momento diferentes partes de la funcionalidad de software requerida generalmente se habrán documentado en diferentes niveles de granularidad.

Método de medición [13]. Una secuencia lógica de operaciones, descrita genéricamente, utilizada en la realización de mediciones.

Patrón de Medición (Estrategia). Una plantilla estándar que puede aplicarse al medir una pieza de software de un dominio funcional de software dado, que define los tipos de usuarios funcionales que pueden interactuar con el software, el nivel de descomposición del software y los tipos de movimientos de datos que el software puede manejar.

Modelo [15]. 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 funcionalmente modificado si se aplica al menos uno de los siguientes:
   - el grupo de datos movido se modifica
   - la manipulación de datos asociada se modifica

b) Se modifica un grupo de datos si se aplica al menos uno de los siguientes:
   - uno o más atributos nuevos se agregan al grupo de datos
   - uno o más atributos existentes se eliminan del grupo de datos
   - se modifican uno o más atributos existentes, p. ej. en significado o formato (pero no en sus valores)

c) Una manipulación de datos se modifica si se cambia funcionalmente de alguna manera.

Requisito No Funcional. Cualquier requerimiento para el software parte de un sistema de hardware/software o producto de software, incluido cómo debe desarrollarse y mantenerse, y cómo debe funcionar en la operación, excepto un requisito del usuario funcional para el software. Los requisitos No Funcionales se refieren a:

- la calidad del software;
- el entorno en el que se debe implementar el software y al que debe servir;
- los procesos y la tecnología que se utilizarán para desarrollar y mantener el software;
- la tecnología que se utilizará para la ejecución del software.

NOTA: Los requisitos del sistema o software que inicialmente se expresan como no funcionales a menudo evolucionan a medida que un proyecto avanza total o parcialmente a FUR para software.

Tipo de objeto de interés. Cualquier "cosa" en el mundo del usuario funcional que se identifica en los Requisitos del Usuario Funcional sobre los cuales se requiere que el software mueva un grupo de datos dentro o fuera del software, o hacia o desde el almacenamiento persistente. Puede ser cualquier cosa física, así como cualquier objeto conceptual o parte de un objeto conceptual.

NOTA 1: En el método COSMIC, el término "objeto de interés" se utiliza para evitar términos relacionados con métodos específicos de ingeniería de software. El término no implica "objetos" en el sentido utilizado en los métodos orientados a objetos.

NOTA 2: Cuando un usuario funcional envía un grupo de datos describiéndose a sí mismo, p. ej. su estado o su identidad, o cuando un usuario funcional recibe datos describiéndose a sí mismo, entonces el usuario funcional también cumple el papel de la "cosa" en su mundo, por lo que también es el objeto de interés descrito por el grupo de datos

NOTA 3: No hay nada absoluto sobre un objeto de interés. Una "cosa" puede ser un objeto "de interés" para un usuario funcional a través de uno o más procesos funcionales, pero no puede ser un objeto "de interés" para otro usuario funcional a través de otros procesos funcionales, incluso en el mismo software que se está midiendo.

salida. Los datos movidos por todas las Salidas de un proceso funcional dado.

Piezas de software iguales. Dos piezas de software son iguales entre sí si residen en la misma capa.

Almacenamiento persistente. Almacenamiento que permite que un proceso funcional almacene datos más allá de la vida útil del proceso funcional y/o que permite que un proceso funcional recupere datos almacenados por otro proceso funcional, o almacenados por una ocurrencia anterior del mismo proceso funcional o almacenados por algún otro proceso.

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

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

Pieza de software. Cualquier elemento discreto de software en cualquier nivel de descomposición desde el nivel de un sistema de software completo hasta el nivel del componente más pequeño de un sistema de software.

Propósito de una medida. Una declaración que define por qué se realiza una medición y para qué se utilizará el resultado.

R. Abreviatura de "Tipo de Lectura".

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

NOTA: Se considera que un tipo de Lectura representa cierta manipulación de datos asociada; consulte el Manual de medición para obtener más información.

Alcance (de una medida). El conjunto de Requisitos del Usuario Funcional que se incluirán en una instancia de medición de tamaño funcional específica. [17]

NOTA: (Específico para el método COSMIC). Se debe hacer una distinción entre el "alcance general", es decir, todo el software que debe medirse de acuerdo con el propósito, y el "alcance" de cualquier pieza de software individual dentro del alcance general. , cuyo tamaño debe medirse por separado. En el Manual de medición, el término "alcance" (o la expresión "alcance de medición") se relacionará con una pieza de software individual cuyo tamaño debe medirse por separado.

Software [17]. Un conjunto de instrucciones de computadora, datos, procedimientos y tal vez documentación que opera en conjunto, para cumplir un conjunto específico de propósitos, todo lo cual puede describirse desde una perspectiva funcional a través de un conjunto finito de Requisitos del Usuario Funcional, requisitos técnicos y de calidad.

Sistema de software. Un sistema que consta solo de software.

Tipo de subproceso. Una parte de un proceso funcional que mueve datos (dentro del software de un usuario funcional o fuera del software a un usuario funcional, o hacia o desde un almacenamiento persistente) o que manipula datos.

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

NOTA: La definición anterior es una adaptación de la definición ISO/IEC 15288: 2008. En la definición COSMIC, “hardware, software y procedimientos manuales” reemplaza “elementos interactuantes” en la definición ISO/IEC.

Tipo de 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 comenzar a procesarse.

NOTA: Esta definición es el resultado del Modelo Genérico de Software, que es un modelo lógico. Físicamente, un proceso funcional puede comenzar a procesarse antes de que se hayan ingresado los datos, p. ej. cuando un usuario humano hace clic en un menú para mostrar una pantalla vacía para la entrada de datos.

Tipo de evento desencadenante. Un evento, reconocido en los Requisitos del Usuario Funcional del software que se está midiendo, que hace que uno o más usuarios funcionales de este software generen uno o más grupos de datos. El primer grupo de datos generado por cualquier usuario funcional será movido posteriormente por una Entrada desencadenante. Un evento desencadenante no se puede subdividir y ha sucedido o no.

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

Unidad de medida [14]. Una cantidad particular, definida y adoptada por convención, con la que se comparan otras cantidades del mismo tipo para expresar sus magnitudes en relación con esa cantidad. Cabe señalar que las unidades de medida tienen nombres y símbolos asignados convencionalmente.

Ver también "Unidad de medida COSMIC"

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

NOTA: Los ejemplos de 'cosa' incluyen, entre otros, aplicaciones de software, animales, sensores u otro hardware.

Valor (de una cantidad) [14]. La magnitud de una cantidad particular, generalmente expresada como una unidad de medida multiplicada por un número.

W. Abreviatura de "Tipo de **Escritura**".

Tipo de Escritura. Un movimiento de datos que mueve un grupo de datos desde el interior de un proceso funcional al almacenamiento persistente.

Nota: se considera que una Escritura representa cierta manipulación de datos asociada; consulte el Manual de medición para obtener más información.

X. Abreviatura de "Tipo de Salida".

APENDICE G – PROCEDIMIENTO DE SOLICITUD DE CAMBIO Y COMENTARIO

El Comité de Prácticas de Medición COSMIC (MPC) está muy ansioso por recibir comentarios, comentarios y, si es necesario, solicitudes de cambio para esta guía. Este apéndice establece cómo comunicarse con COSMIC MPC. Todas las comunicaciones al COSMIC MPC deben enviarse por correo electrónico a la siguiente dirección:

mpc-chair@cosmic-sizing.org

Retroalimentación y comentarios generales informales

Los comentarios informales y/o retroalimentación sobre las guías, tales como cualquier dificultad para comprender o aplicar el método COSMIC, sugerencias para mejoras generales, etc., deben enviarse por correo electrónico a la dirección anterior. Los mensajes se registrarán y generalmente se acusarán recibo dentro de las dos semanas posteriores a la recepción. El MPC no puede garantizar la acción de tales comentarios generales.

Solicitudes formales de cambio

Cuando el lector de la guía cree que hay un defecto en el texto, una necesidad de aclaración o que algún texto necesita mejorarse, se puede enviar una Solicitud de Cambio Formal ("CR"). Los CR formales se registrarán y se confirmarán dentro de las dos semanas posteriores a la recepción. A cada CR se le asignará un número de serie y se distribuirá a los miembros del COSMIC MPC, un grupo mundial de expertos en el método COSMIC. Su ciclo de revisión normal toma un mínimo de un mes y puede tomar más tiempo si la CR resulta difícil de resolver. El resultado de la revisión puede ser que la CR sea aceptada, rechazada o “retenida en espera de una discusión adicional” (en este último caso, por ejemplo, si existe una dependencia de otra CR), y el resultado se comunicará nuevamente a el remitente tan pronto como sea posible.

Se aceptará un CR formal solo si está documentado con toda la siguiente información.

- Nombre, cargo y organización de la persona que presenta el CR.
- Datos de contacto de la persona que presenta el CR.
- Fecha de presentación.
- Declaración general del propósito de la RC (por ejemplo, "necesita mejorar el texto ...").
- Texto real que necesita ser cambiado, reemplazado o eliminado (o una referencia clara al respecto).
- Texto adicional o de reemplazo propuesto.
- Explicación completa de por qué es necesario el cambio.

Un formulario para enviar un CR está disponible en la base de conocimiento de www.cosmic-sizing.org. La decisión del COSMIC MPC sobre el resultado de una revisión de RC y, si se acepta, en qué versión se aplicará el CR, es definitiva.

Preguntas sobre la aplicación del método COSMIC

Puede usar el foro en cosmic-sizing.org/forums para publicar sus preguntas y recibir respuestas de nuestra comunidad mundial. La calidad de cualquier respuesta dependerá del conocimiento y la experiencia del miembro de la comunidad que escribe la respuesta; El MPC no puede garantizar la corrección. Existen organizaciones comerciales que pueden proporcionar capacitación y consultoría o soporte de herramientas para el método. Por favor, consulte el sitio web www.cosmic-sizing.org para más detalles.