Metodologías modernas de ingeniería de software para ambientes móviles y en la nube

Antonio Miguel Rosado da Cruz

Instituto Politécnico de Viana do Castelo, Portugal

Sara Paiva

Instituto Politécnico de Viana do Castelo, Portugal

Un volumen de la serie de libros “Avances en el análisis de sistemas, ingeniería de software y cómputo de alto desempeño” (ASASEHPC)

Information Science REFERENCE
An Imprint of IGI Global

Publicado en los Estados Unidos de América por

Information Science Reference (una impresión de IGI Global)
701 E. Chocolate Avenue
Hershey PA, USA 17033
Tel: 717-533-8845
Fax: 717-533-8661
Correo electrónico: cust@igi-global.com
Página web: http://www.igi-global.com

Librería del congreso. Catalogación en la publicación de datos

Nombres: Cruz, Antonio Miguel Rosado da, 1970- editor. | Paiva, Sara, 1979- editor.

Título: Metodologías modernas en ingeniería de software para ambientes móviles y en la nube / Antonio Miguel Rosado da Cruz and Sara Paiva, editores.

Descripción: Hershey, PA: Information Science Reference, 2016. | Incluye referencias bibliográficas e índice.

Identificadores: LCCN 2015046896| ISBN 9781466699168 (pasta dura) | ISBN 9781466699175 (libro electrónico)

Temas: LCSH: Cómputo en la nube. | Cómputo móvil. | Ingeniería de software.

Clasificación: LCC QA76.585 .M645 2106 | DDC 004.67/82--dc23 LC registro disponible en http://lccn.loc.gov/2015046896

Este libro está publicado en la serie de libros IGI Global: Avances en análisis de sistemas, ingeniería de software, y cómputo de alto desempeño (ASEHPC) (ISSN: 2327-3453; eISSN: 2327-3461)

Datos de la catalogación en la publicación británica
El registro de catalogación en publicación para este libro está disponible en la Librería Británica.

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

Capítulo 4

Estimación para ambientes móviles y en la nube

Frank Vogelezang

Ordina, Holanda

Jayakumar Kamala Ramasubramani

Amitysoft Technologies, India

Srikanth Arvamudhan

Amitysoft Technologies, India

RESUMEN

La estimación del costo, esfuerzo y calendario es un aspecto muy importante en el desarrollo del software comercial. El esfuerzo es, por lo general, la unidad de costo predominante en el desarrollo del software. El determinante dominante para el esfuerzo es el tamaño del software por desarrollarse. Existen varias formas de determinar el tamaño del software. La mejor opción es utilizar una métrica estandarizada para obtener el tamaño funcional. En este capítulo se introduce el método COSMIC para obtener el tamaño funcional. Debido a sus principios básicos, el método COSMIC permite determinar el tamaño funcional de aplicaciones móviles y en la nube. Este capítulo demuestra cómo es que el método COSMIC proporciona una buena base para estimar el costo, esfuerzo y calendario en los ambientes móviles y en la nube.

INTRODUCCIÓN

La estimación del costo, esfuerzo y calendario es un aspecto muy importante en el desarrollo del software comercial, ya que se utiliza para asignar recursos, planificar versiones del producto y negociar pagos entre proveedores y contratistas. En el campo de conocimiento de la ingeniería de software, se especifica a la estimación como una subárea Planeación de Proyecto para el área de conocimiento de la administración en la ingeniera de software (Sociedad de cómputo de la IEEE, 2014). En el desarrollo del software, el esfuerzo es, por lo general, la unidad de costo predominante. Sin embargo, la estimación del costo y del calendario, depende de la estimación del esfuerzo. El determinante dominante para el esfuerzo es el tamaño del software a desarrollarse.

DOI: 10.4018/978-1-4666-9916-8.ch004

Existen varias maneras de definir el tamaño, que pueden dividirse en medidas de tamaño técnicas y funcionales. Las medidas de tamaño técnicas son fáciles de obtener del mismo software, pero se ha demostrado que no son predictores muy precisos para estimar el esfuerzo (Jones, 2013). En el desarrollo del software tradicional se han desarrollado varios métodos para medir el tamaño funcional llamados de "primera generación" como son el Análisis de Puntos de Función, Análisis de Puntos Característicos y Puntos por Caso de Uso (Vogelezang, 2013a). Estos métodos fueron desarrollados de manera empírica y se basaron en el paradigma de aplicaciones autónomas con interacciones limitadas con otro software. Este base de paradigma no es la adecuada para el desarrollo de software en móviles y en la nube, por su arquitectura en capas e interacción multinivel con otros componentes del software.

El método COSMIC es un método para medir el tamaño funcional de segunda generación que se basa en la idea de que el software puede existir en múltiples capas, y que los datos que se mueven de un componente a otro del software son el predictor principal para obtener el tamaño funcional del software (Vogelezang, 2013b). Se reconoce cada vez más al método COSMIC por su aplicabilidad para estimar el esfuerzo de desarrollo en soluciones móviles y en la nube como las aplicaciones de negocios multicapas y orientadas al servicio.

ANTECEDENTE: TAMAÑO FUNCIONAL

Durante años, las organizaciones dedicadas a la ingeniería de software se han esforzado en encontrar métodos cuantitativos aceptables para medir la eficacia y la efectividad de los procesos y para administrar los costos de software, para los sistemas que adquieren, desarrollan, mejoran o mantienen. Un aspecto crítico y particularmente elusivo de este requerimiento de medición ha sido la necesidad de determinar el tamaño del software. En 1998, el concepto de tamaño funcional se definió con una norma internacional (ISO, 2007).

El tamaño funcional es una medida de la cantidad de funcionalidad proporcionada por el software, que se obtiene al cuantificar las tareas y procedimientos que el software debe realizar para cumplir las necesidades del usuario, independiente de cualquier consideración técnica o de calidad. El tamaño funcional es, por lo tanto, una medida de lo que el software debe hacer, y no del cómo debería funcionar. Esto significa que el tamaño funcional puede determinarse antes de que se construya el software real y aun antes de que se decida en que plataforma y en que lenguaje de programación se construirá.

El concepto de tamaño funcional es, por lo tanto, una base ideal de comparación, ya sea para comparar software con características similares o para comparar plataformas. Estos diferentes tipos de comparación pueden ser útiles para evaluar el desarrollo y mantenimiento o como apoyo en la elección de una cierta plataforma.

Medición del tamaño funcional

Medir el tamaño funcional significa medir la funcionalidad que se desarrollará en función de los requerimientos del software. Los requerimientos describen la funcionalidad, y ciertas características de esa funcionalidad, que se construirán con el fin de entregar el software adecuado para cierto propósito. ISO ha desarrollado una metanorma para los métodos de medición del tamaño funcional (ISO, 2007). Todos los métodos para medir el tamaño funcional deben cumplir con esta norma. En el 2003, ISO reconoció al método COSMIC, que es totalmente compatible con la metanorma, como un estándar para medir el tamaño funcional (ISO, 2011).

Aunque el objetivo principal del método COSMIC es medir el tamaño funcional, se efectuó una prueba de campo limitada a aplicaciones en tiempo real para verificar si el método COSMIC era adecuado para predecir el esfuerzo en la especificación, construcción y prueba antes de presentarlo como una Norma Internacional (Abran, 2001). Recientemente, el International Software Benchmarking Standards Group (ISBSG, por sus siglas en inglés) analizó los datos de rendimiento de 324 proyectos de software de aplicaciones de negocios, 40 proyectos de software en tiempo real y 22 proyectos que produjeron componentes de software (ISBSG, 2012). Su conclusión fue que los datos de referencia COSMIC fueron consistentes para todos los tipos de proyectos analizados y adecuados para las comparaciones externas de desempeño y para estimar nuevos proyectos. Al comparar los datos con los datos de referencia del IFPUG, con el método COSMIC se obtuvieron cifras más diferenciadas para diferentes conjuntos de características del proyecto.

El método COSMIC

Con el método COSMIC se transformaron los métodos de primera generación con el uso de un nuevo paradigma y de principios de ingeniería de software establecidos. El método COSMIC es un método de medición de tamaño funcional de segunda generación que se basa en la idea de que el software puede existir en múltiples capas, y que mover datos de un componente de software a otro es el principal predictor del tamaño funcional del software. El nuevo paradigma asegura que el método se puede aplicar a todo el software donde el movimiento de datos es el principal predictor para obtener el tamaño funcional. Esto significa que el método COSMIC se puede aplicar tanto al software de aplicaciones de negocios como al software en tiempo real, lo que lo convierte en un recurso de estimación muy potente en el dominio móvil y en la nube. El método COSMIC es mantenido por el Common Software Measurement International Consortium.

EL MODELO DE SOFTWARE COSMIC

El método COSMIC considera al software como un conjunto de *procesos funcionales* basados en los requerimientos funcionales del usuario. A los usuarios que interactúan con software cruzando fronteras a través de procesos funcionales se les llama *usuarios funcionales*. Los procesos funcionales hacen dos cosas: mueven *grupos de datos* hacia y desde el proceso funcional y manipulan grupos de datos dentro del proceso funcional. Se supone que la manipulación de datos se explica por el movimiento de los grupos de datos asociados con la manipulación.

El movimiento de un solo y único grupo de datos hacia o desde el proceso funcional se denomina *movimiento de datos*. Con base en la funcionalidad del proceso, los grupos de datos

se mueven entre 'el usuario y el software', así como entre el 'software y el almacenamiento persistente'. El método COSMIC distingue cuatro tipos de movimientos de datos (COSMIC, 2015a).

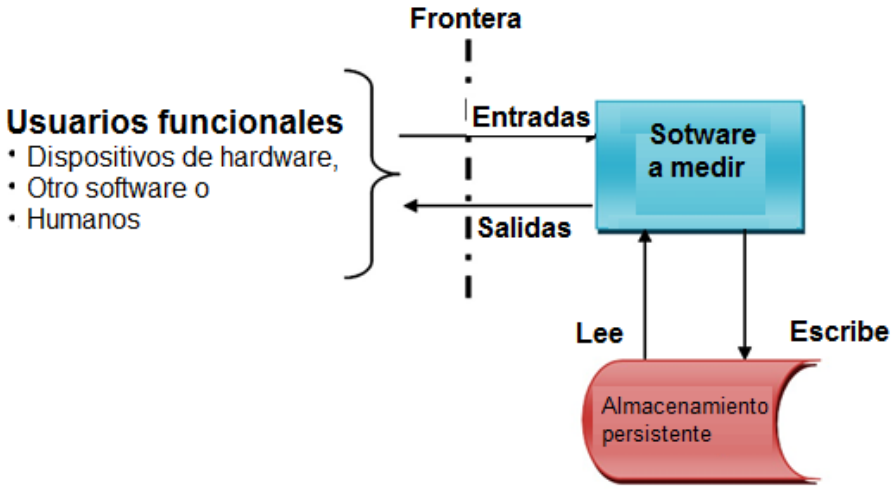
- El movimiento de datos de *entrada* mueve datos de los usuarios funcionales al software
- El movimiento de datos de *salida* mueve datos fuera del software a los usuarios funcionales
- El movimiento de datos de *lectura* mueve datos del almacén de datos al software
- El movimiento de datos de *escritura* mueve datos del software al almacén de datos

En la figura 1 se describe lo anterior.

La medición del tamaño funcional COSMIC se basa en el número de movimientos de datos relacionados con los procesos funcionales ejecutados por los usuarios (figura 2).

La suma de todos los movimientos de datos asociados con todos los procesos funcionales asociados con el software es el tamaño de software en *Puntos de Función COSMIC*, o bien *CFP*.

Figura 1. Los cuatro tipos de movimientos de datos



El proceso de medición COSMIC

Con el fin de determinar el tamaño funcional se necesita un proceso repetible: El método COSMIC tiene un proceso de medición muy bien descrito (COSMIC, 2015a), que consiste de tres pasos principales:

Primero, se debe definir lo que se quiere medir. Por lo tanto, debe haber un acuerdo entre el propósito y el alcance de la medición. Esto determina quién, o qué está definido como usuarios funcionales del software. El siguiente paso es transformar la descripción funcional en procesos funcionales y movimientos de datos del modelo de software genérico COSMIC.

El último paso es la medición real. El tamaño del software se determina con la identificación de todos los movimientos de datos (entradas, salidas, lecturas y escrituras) de

cada proceso funcional y agregando el número de movimientos de datos de todos los procesos funcionales.

Figura 2. Modelo de software genérico COSMIC

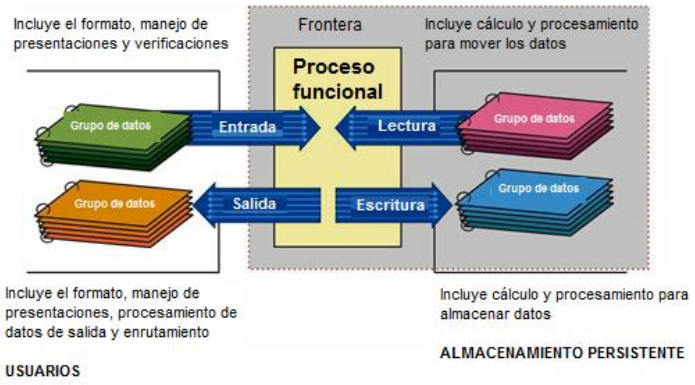
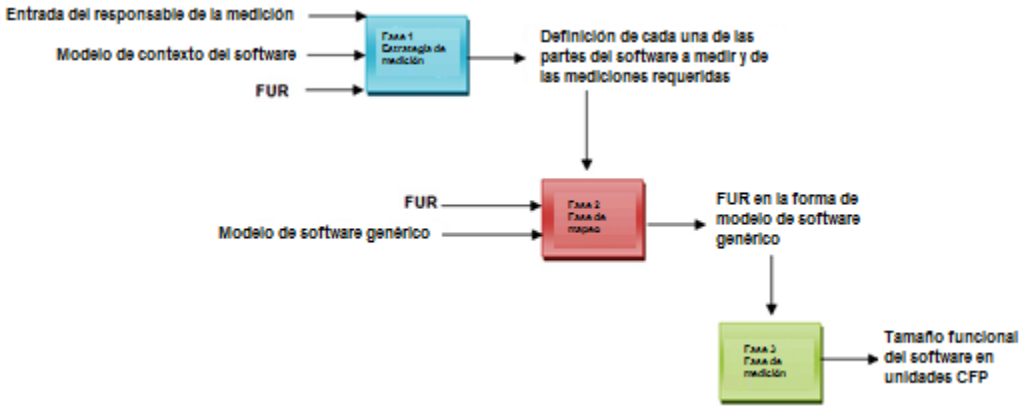


Figura 3. Proceso de medición COSMIC



En la figura 3 se muestran los pasos del proceso de medición COSMIC con entradas y salidas de cada una de las etapas.

El concepto de usuario funcional en COSMIC

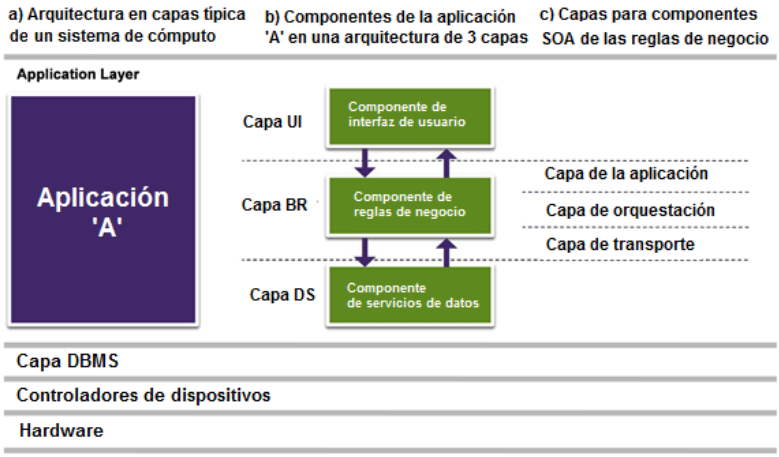
Una de los conceptos clave del método COSMIC es el usuario funcional (COSMIC, 2015a). Los usuarios funcionales pueden ser seres humanos, software o dispositivos de ingeniería que interactúan con el software a medir. En general, los usuarios funcionales pueden identificarse como aquellos que envían y/o reciben datos a, y desde, el software. Estos pueden ser seres humanos, dispositivos de ingeniería y componentes pares de la misma plataforma o aplicaciones externas en una plataforma diferente (figura 4).

Los diferentes tipos de usuarios pueden requerir diferente funcionalidad y, por lo tanto, los tamaños funcionales variarán de acuerdo a los usuarios funcionales. Esta es la razón por la cual la estrategia de medición es un paso importante en el proceso de medición. Por lo tanto,

la medida COSMIC se basa en los requerimientos de usuario funcionales (FUR, por sus siglas en inglés) y no en los artefactos u objetos de la implementación. Aunque COSMIC es el único método que describe este proceso, éste puede aplicarse a todo tipo de procesos de medición.

Figura 4. Usuarios funcionales de software

Figure 5. Tres vistas de capas



El concepto de capa en COSMIC

COSMIC es el único método de medición de tamaño funcional que reconoce las capas de acuerdo con los estándares de la arquitectura de software. Los requerimientos funcionales pueden mapearse a cualquier capa con base en la naturaleza de los servicios que ofrece. La figura 5 proporciona tres vistas de capas como lo reconoce (COSMIC, 2015 a).

En la primera vista, se considera a la aplicación como monolítica, que contiene toda la funcionalidad, con excepción del sistema de administración de base de datos, los controladores de dispositivos y el sistema operativo. Este tipo de arquitectura se ha vuelto muy raro y generalmente utilizado para soluciones de pequeños negocios de software "rápidos y triviales", como son los programas MS Access temporales.

La arquitectura de software más común para el mundo que se basa en redes, que utiliza la Internet como columna vertebral, es la arquitectura de tres capas en el que la lógica de proceso funcional, el acceso a los datos, el almacenamiento de datos calculados y la interfaz de usuario son desarrollados y mantenidos como módulos independientes en plataformas separadas. Crear capas es un patrón de diseño de software además de una arquitectura de software bien establecida (Janssen, 2014) para poder construir software complejo. Esta arquitectura permite que cualquiera de las capas pueda ser actualizada o reemplazada de manera independiente.

- El componente de interfaz de usuario se implementa en el dispositivo del usuario y utiliza la interfaz gráfica de usuario estándar de ese dispositivo.
- El componente lógico de negocios puede estar ya sea en el dispositivo del usuario (esto es muy común en las aplicaciones móviles) o en un servidor de la aplicación de una solución en la nube o sistema administrativo de negocios.
- El componente de servicio de datos contiene la lógica de almacenamiento de datos de cómputo y, por lo general, está ubicado en un servidor de datos en la nube o como una parte separada de un sistema administrativo de negocios.

En la arquitectura orientada a servicios, la lógica de negocios se divide por los servicios que realiza cada tarea específica. La misma capa de lógica de negocios se divide en múltiples capas. La arquitectura orientada a servicios adoptada en el ambiente en la nube utiliza este paradigma (COSMIC, 2010).

- La capa de aplicación proporciona operaciones específicas y limitadas como servicios. Estos servicios se pueden relacionar como un mecanismo de solicitud/respuesta que alimenta a la interfaz de usuario o capa de orquestación con funcionalidad reutilizable. Ejemplos comunes de la capa de aplicación son: los servicios para crear una reserva o servicios para verificar la autenticación o autorización del usuario funcional del componente de la interfaz de usuario.
- La capa de orquestación llama y controla otros servicios para realizar un proceso completo (negocio). Maneja la comunicación con múltiples servicios de la aplicación y/o servicios del negocio que procesan una parte del proceso (negocio). Ejemplos de una orquestación son: un análisis de riesgo en una aplicación de hipotecas o una ruta de viaje con base en la ubicación desde una aplicación de transporte público.
- La capa de servicios proporciona una funcionalidad común (negocio o de no negocio) de manera independiente, pero disponible, a otras capas o aun en otras aplicaciones. Ejemplos comunes de servicios en la capa de servicios son el registro y análisis de archivo API.

La mayoría de los métodos actuales de medición de tamaño funcional de primera generación sólo pueden tratar con la primera vista que presenta la figura anterior, donde se considera a la aplicación un monolito que contiene toda la funcionalidad. Esto los hace menos poderosos para estimar software móvil o en la nube. Por lo general, el software en la nube o móvil se construyen con un patrón de diseño de acuerdo a la segunda o tercera vista. Las diferentes capas son, por lo general, desarrolladas y mantenidas como módulos independientes en plataformas separadas por equipos separados, de modo que necesitan estimarse por separado (Janssen, 2014).

En COSMIC, el tamaño y estimación se pueden realizar en una capa específica y, por lo tanto, se obtiene una medición más precisa y útil. Como la tecnología para la implementación

de diferentes capas puede variar, es posible utilizar los factores de productividad en cuestión para medir el esfuerzo de desarrollo con más precisión en vez de utilizar un sólo factor de productividad para toda la aplicación.

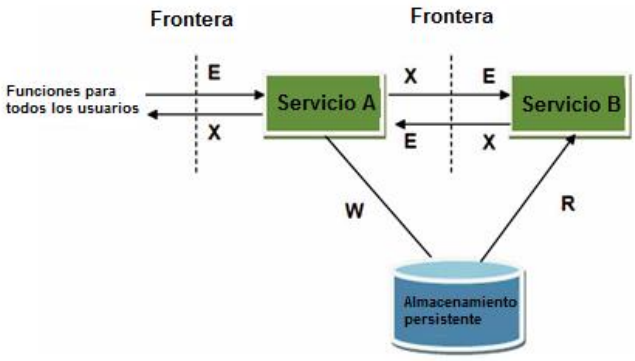
El método COSMIC cuenta con una guía que proporciona un mapeo de los principios de COSMIC con la arquitectura orientada a servicios. Esta guía describe cómo es que se pueden medir los servicios y sus interacciones para determinar el tamaño funcional (COSMIC, 2010). El método COSMIC es capaz de medir los procesos funcionales que residen en las diferentes capas, pero utilizan los mismos datos almacenados, como se muestra en la figura 6. Esto es un concepto que no está presente en cualquiera de los métodos de tamaño funcional de primera generación, pero que es muy común en el software en la nube o móvil.

APLICACIÓN DEL MÉTODO COSMIC PARA AMBIENTES MÓVILES

Características de cómputo móvil

El funcionamiento del software en un ambiente móvil puede clasificarse como híbrido, es decir, es tanto un tipo de aplicación en tiempo real como de negocios. El sistema operativo y los componentes críticos se ejecutan en tiempo real como cualquier otro sistema en tiempo real. Las aplicaciones móviles que se construyen en la parte superior se comportan como aplicaciones de negocios con características específicas para un ambiente móvil. Aunque las aplicaciones móviles se pueden considerar como un tipo moderno de arquitectura cliente-servidor, son, de muchas maneras, diferentes a las aplicaciones tradicionales (van Heeringen & van Gorp, 2014).

Figura 6. Mapeo de los movimientos de datos COSMIC con una arquitectura orientada a servicios



- El usuario puede interactuar con las aplicaciones de más maneras que en las aplicaciones tradicionales. Por ejemplo, la aplicación puede activarse al cambiar de posición (invertir) el dispositivo móvil, agitar el dispositivo móvil y algunas aplicaciones pueden aceptar mensajes de voz.

- También algunas aplicaciones responden a eventos en tiempo real, como al mover el dispositivo móvil, cambiar de Wi-Fi a modo de datos celulares o cuando la red está fuera del alcance.
- Las aplicaciones deben tener funcionalidad para maneja interrupciones, como, por ejemplo, cuando entra una llamada.
- Existen, por lo general, requerimientos importantes no funcionales que la aplicación debe cumplir, p. ej., seguridad, desempeño, tráfico mínimo de datos, espacio ocupado en el dispositivo y consumo de batería.

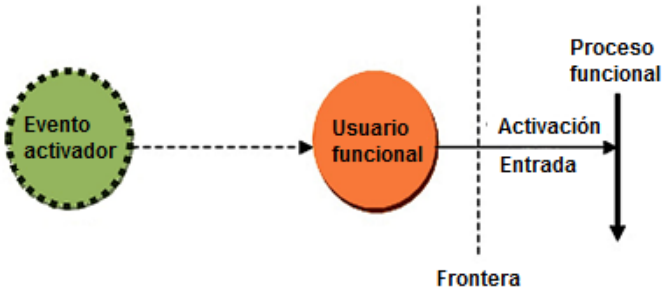
Contexto de la arquitectura móvil

El concepto de usuario funcional en COSMIC incluye a todas las interacciones de los usuarios de todo tipo presentes en un ambiente móvil. Los usuarios funcionales pueden ser usuarios humanos, dispositivos de ingeniería, y componentes pares de la misma plataforma, o aplicaciones externas en una plataforma diferente.

En el método COSMIC todos los procesos funcionales responden a eventos en el mundo real. Un usuario funcional detecta un evento. El usuario funcional activa un proceso funcional. Cuando un sensor detecta una condición en el que el software debe responder, el sensor se convierte en el usuario funcional y el evento activado es la condición con la que el sensor está diseñado a detectar. El movimiento de datos de entrada activado es el mensaje del sensor al software anunciando que el evento ha ocurrido. El mensaje también puede llevar datos del evento activado. Esto se describe en la figura 7 (COSMIC, 2015a).

Mientras que los procesos en el software de aplicaciones de negocios involucran sobre todo entradas y salidas de datos, hay una cantidad significativa de procesamiento interno, en el caso de aplicaciones móviles, que no está directamente relacionado con el procesamiento de entradas y salidas, como la visualización de gráficos, imágenes o resaltar ciertos datos, con base en reglas de procesamiento. Cuando se utilizan métodos de primera generación como IFPUG FPA, no se acreditan como funcionalidad entregada aquellos procesos que cuentan con una cantidad significante de operaciones internas pero que tienen poca visibilidad externa.

Figura 7. La relación entre eventos, usuarios funcionales y procesos funcionales



Para las aplicaciones móviles, el almacenamiento y mantenimiento de datos no es tan importante como en el software de aplicaciones de negocios. Estas aplicaciones se conectan a la nube o a los sistemas back-office (de administración) y recuperan solo los resultados del procesamiento de datos. Por lo general, en los procesos internos se utiliza información en tiempo real para poder presentar los datos recuperados en el contexto correcto. La entrada de datos al proceso no siempre se almacena de forma permanente, sino que se recuperan externamente y se utilizan durante el proceso que requiere los datos en cuestión. Incluso cuando se trata de aplicaciones que no son en tiempo real, como son las aplicaciones de negocios, algunos datos pueden almacenarse en el dispositivo móvil, pero la mayoría de éstos estarán en la nube o en servidores de back-office. La arquitectura exacta puede variar para cada aplicación, pero como regla general, para las aplicaciones móviles. Los datos almacenados no son tan importantes.

Los métodos de medición de tamaño funcional de primera generación dependen en gran medida de la funcionalidad de los datos almacenados y mantenidos. No es posible mapear estos métodos de primera generación, como son los puntos de función IFPUG, a una arquitectura móvil, de una manera natural, en donde el concepto de archivos internos y externos tienen que identificarse de manera explícita. El método COSMIC se ajusta directamente a las necesidades de dimensionamiento y estimación en el entorno móvil para desarrollar componentes en tiempo real y aplicaciones de negocios. El método COSMIC sólo tiene que saber que hay una fuente de datos en algún lugar para recuperar los datos. Esta idea se muestra en la figura 8.

En una aplicación móvil, los usuarios humanos se comunican con la aplicación mediante los movimientos de datos de entrada y salida. Por lo general, la aplicación también se comunicará en tiempo real con los componentes pares del dispositivo móvil a través de los movimientos de entrada y salida de datos. Si los datos necesitan almacenarse en el dispositivo, la aplicación se comunicará con el almacenamiento permanente mediante los movimientos de lectura y escritura de datos. La aplicación puede comunicarse con un sistema administrativo *back-end* o con un servicio en la nube a través de los movimientos de entrada y salida de datos

Ejemplo de una aplicación bancaria

Para ilustrar como se aplica el método COSMIC, mostramos el funcionamiento de una aplicación bancaria común (figura 9) y la descripción de sus transacciones. Esto es una vista básica del proceso funcional:

E Entrada activadora
X Solicitar datos de transacción al Orquestador
E Recibir los datos de transacción
E Recibir de los mensajes de error del Orquestador
X Mostrar transacciones
X Mostrar mensajes de error de la aplicación

Figura 8. Ejemplo de movimientos de datos en una aplicación móvil

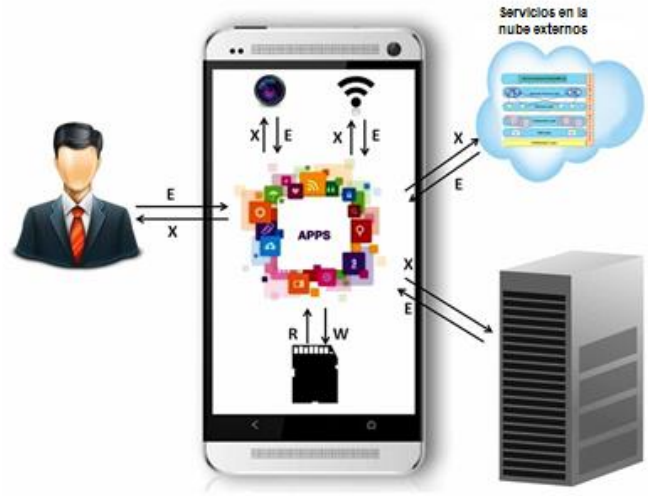


Figura 9. Aplicación bancaria

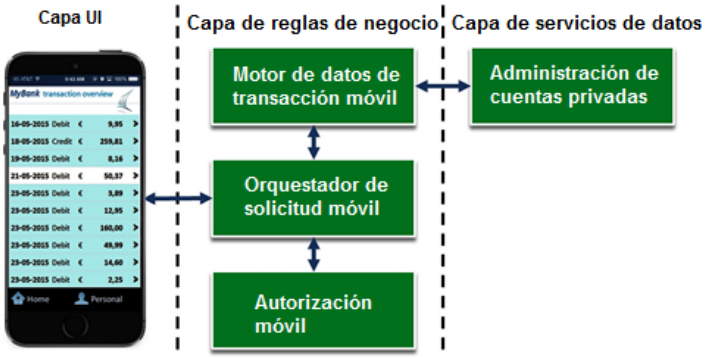
Este proceso funcional consta de seis movimientos de datos y, por lo tanto, el tamaño funcional de la descripción de la transacción es de 6 CFP. Este proceso funcional incluye la vista de todos los atributos de datos de un grupo de datos que pertenece a un objeto de interés. Al inclinar el dispositivo móvil se muestran más atributos de datos. Esto se considera que es la visualización del mismo grupo de datos y no significa un nuevo proceso funcional. La inclinación del dispositivo se considera un comando de control dentro de este proceso funcional que no se cuenta como un movimiento de datos por separado. Para la medición del tamaño funcional esto es similar a la capacidad de cambiar la visión de los datos desplegados (p. ej., al filtrar o clasificar los datos), lo que tampoco significan movimientos de datos adicionales (van Heeringen & van Gorp, 2014).

El resumen de las transacciones representa solamente la funcionalidad de la aplicación. Poderle presentar al usuario de esta aplicación sus transacciones financieras, no es la única funcionalidad que se necesita desarrollar para que funcione la aplicación. Como se indicó anteriormente, el tamaño funcional COSMIC se cuenta por cada capa de la arquitectura. En la siguiente figura (figura 10) se muestra la parte relevante de toda la arquitectura de la aplicación bancaria de este caso.

La aplicación envía la solicitud al componente Orquestador de Solicitudes Móviles (Mobile Request Orchestrator) en la capa de Reglas de Negocio. El orquestador verifica con el componente Autorización del Móvil (Authorization Mobile) que el dispositivo móvil está autorizado para recibir transacciones financieras. Cuando el dispositivo está autorizado, el orquestador solicita los datos de transacción del Motor de Datos de Transacciones Móviles (Mobile Transaction Data Engine) y los devuelve a la aplicación. Cuando se genera un error en la autorización, o no hay datos de transacción disponibles, el Motor de Datos de Transacciones Móviles realiza una copia exacta de las transacciones financieras en la Administración de Cuentas Privadas (Private Accounts Administration) que sólo guarda los atributos de datos que se utilizarán en los dispositivos móviles. La Administración de Cuentas Privadas no es capaz de atender el número de solicitudes desde los dispositivos móviles, por lo que el componente Motor de Datos se introduce en la arquitectura del banco.

Para el resumen de las transacciones, se necesitan los procesos funcionales de todos los componentes mostrados en figura anterior:

Figura 10. Elementos esenciales de la arquitectura bancaria que soportan la aplicación móvil



Orquestador de solicitudes móviles

Generar el proceso funcional del resumen de las transacciones

E Solicitar datos de transacción desde el resumen de transacciones (*Entrada* activadora)
X Solicitar autorización al componente Autorización Móvil
E Recibir la información de Autorización Móvil
E Recibir mensaje de error de Autorización Móvil
X Solicitar datos de transacción al Motor de Datos de Transacciones Móviles
E Recibir datos de transacción del Motor de Datos de Transacciones Móviles
E Recibir mensaje de error de Datos de Transacciones Móviles
X Enviar datos de transacción a la aplicación
X Enviar mensajes de error a la aplicación
9 CFP

Autorización del móvil

Autorizar el proceso funcional del dispositivo solicitante

E Solicitar autorización al Orquestador de Solicitud Móvil (*Entrada* activadora)
R Verificar si el dispositivo solicitante está autorizado
W Almacenar fecha y hora de la solicitud de autorización *(si se autoriza*)
X Enviar un mensaje de autorización al Orquestador de Solicitud Móvil
X Enviar mensaje de error al Orquestador de Solicitud Móvil
5 CFP

Motor de Datos de Transacciones Móviles

Proceso funcional de resumen de transacciones

E Solicitar datos de transacción desde el Orquestador de Solicitud Móvil (*Entrada* activadora)
R Recuperar datos de transacción del Motor de Datos
W Almacenar fecha y hora de la última transacción recuperada
X Enviar datos de transacciones al Orquestador de Solicitud Móvil
X Enviar mensaje de error al Orquestador de Solicitud Móvil
5 CFP

Motor de Datos Transacciones Móviles

Proceso funcional de actualización de datos de transacciones

E Actualizar datos de transacción de la capa de Servicio de Datos (*Entrada* activadora)
W Almacenar datos de transacción en el Motor de Datos
W Almacenar fecha y hora de la última transacción recuperada
X Enviar mensaje de finalización (error u OK) a la capa de Servicio de Datos
4 CFP

Administración de cuentas privadas

Proceso funcional de la copia de extracción de datos de transacciones

E Mensaje de finalización de la capa de Reglas de Negocio (*Entrada* activadora)
R Leer la última fecha y hora (en caso de OK) o la anterior (en caso de error)
R Recuperar la transacción después de la última fecha y hora para la copia de extracción
W Almacenar la fecha y hora de la última transacción recuperada
X Enviar copia de extracción de datos de la transacción a la capa de Reglas de Negocio
5 CFP

Por lo tanto, el proceso funcional de 6 CFP del resumen de las transacciones en la aplicación, utiliza cuatro procesos funcionales en la capa de Reglas de Negocio de 23 CFP en total y un proceso funcional en la capa de Servicios de Datos de 5 CFP. Las medidas de tamaño funcional de diferentes capas nunca deben combinarse (COSMIC, 2010).

Este ejemplo muestra que es importante conocer si se encuentra disponible la infraestructura para proporcionar los datos necesarios para la aplicación. La funcionalidad de la aplicación es sólo 6 CFP, pero la infraestructura en la capa de Reglas de Negocio y la capa de Servicios de Datos agregan 23 y 5 CFPs, respectivamente. Esto no sólo incrementa el tamaño funcional, sino que las diferentes capas requieren prácticas diferentes para implementar la funcionalidad.

El ejemplo funciona con una arquitectura de 3 capas (figura 5b) para poder cumplir con los requerimientos para el volumen de transacciones. En el caso de que se hubiera elegido una arquitectura SOA (figura 5c), el proceso funcional *Generar Resumen de Transacciones* habría estado en la capa de Orquestación, el proceso funcional *Resumen de Transacciones* en la capa de Aplicación y los procesos funcionales de *Solicitar Autorización del Dispositivo* y *Actualizar Datos de la Transacción* en la capa de Servicios.

Esto muestra que la elección de una estructura de capas no influye en el tamaño funcional total, sino sólo en la distribución sobre las diferentes capas. Esto no solo es relevante para los ambientes móviles y también para los ambientes en la nube, porque al observar la arquitectura completa, se puede ver que existen grandes coincidencias entre el ambiente móvil mostrado en el ejemplo y una arquitectura básica en la nube (véase también la figura 8).

APLICACIÓN DEL MÉTODO COSMIC PARA AMBIENTES EN LA NUBE

Características de cómputo en la nube

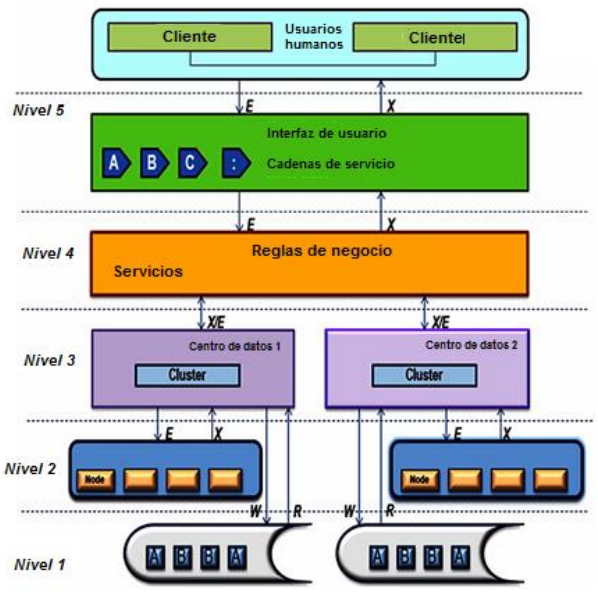
La computación en la nube es un modelo que permite el acceso a una red omnipresente y conveniente bajo demanda a un conjunto compartido de recursos de cómputo configurables (p. ej., redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden abastecer y liberar rápidamente con un esfuerzo de administración o interacción del proveedor de servicios. Este modelo en la nube está compuesto de cinco características esenciales, tres modelos de servicio y cuatro modelos de implantación. (NIST, 2011). Normalmente, los recursos de la nube no sólo son compartidos por múltiples usuarios, sino que también se reasignan, de manera dinámica, bajo demanda. Con la computación en la nube, varios usuarios pueden tener acceso a un único servidor para recuperar y actualizar sus datos sin necesidad de adquirir las licencias para diferentes aplicaciones. La computación en nube permite a las empresas poner

a funcionar sus aplicaciones con mayor rapidez, con una capacidad de administración mejorada y menor mantenimiento, y permite a la TI ajustar con rapidez los recursos para satisfacer la demanda empresarial que, por lo general, es fluctuante e impredecible. La disponibilidad actual en las redes de alta capacidad, computadoras de bajo costo y dispositivos de almacenamiento fueron las condiciones previas esenciales que han llevado a la aparición de la computación en la nube (figura 11).

El ambiente en la nube se caracteriza por lo siguiente:

- Acceso a redes de alta velocidad que promueve el uso de plataformas heterogéneas.
- Agrupación de recursos para crear economías de escala.
- Un arrendamiento múltiple que permite compartir recursos y costos a través de un grupo grande de usuarios.
- Autoservicio bajo demanda que no requiere de la interacción humana.
- Elasticidad que permite que las aplicaciones escalen rápidamente, con base en la demanda real de servicio.
- Confiabilidad mediante el uso de múltiples sitios, proporcionando continuidad de negocio y una fácil recuperación de desastres.
- Servicio medido que permite transparencia en el gasto de TI con base en el uso real.
- Puede vincularse a grandes repositorios de datos (*Data Warehouse*) para grandes aplicaciones de datos.

Figura 11. Modelos de medición COSMIC para software en la nube



La pila (*stack*) de cómputo del software en la nube se compone de los siguientes paradigmas de software:

- Virtualización, la principal tecnología habilitadora para hacer que el software en la nube sea independiente de las plataformas de hardware reales detrás.
- Arquitectura orientada a servicios, para dividir los procesos de negocio en servicios que pueden conectarse y consumirse según se necesiten.
- APIs programables, que ofrecen la capacidad de interactuar con los servicios en la nube disponibles, independientemente del dispositivo de conexión del usuario (PC, tableta o teléfono móvil).
- Capa de administración, que proporciona información en tiempo real sobre el uso actual de los servicios de administración y facturación.

En 2013, investigadores alemanes propusieron una arquitectura que describe los diferentes niveles de funcionalidad involucrados en el software en la nube (Schmietendorf, 2013). Esta Arquitectura de Sistemas en la Nube consta de cinco niveles de funcionalidad:

Nivel 1: representa las diferentes máquinas (ya sea virtuales o no) que ofrecen un servicio. Estas máquinas tienen cierta prioridad de acuerdo a la criticidad de la interrupción.

Nivel 2: mapea el factor de agrupación en un nivel de máquina, aunque no todas las máquinas y tipos de rol son necesariamente agrupables. En estos dos niveles pueden ocurrir diferentes dificultades de eventos e incidentes (registro, repetición, información de la capacidad, etc.)

Nivel 3: consolida el nivel del centro de datos que representa los aspectos de las redundancias múltiples a nivel de alojamiento físico (como los enfoques de centro de datos de doble o triple núcleo) en donde se ofrecen servicios en redundancia completa. Si un centro de datos está lleno, o algún servicio alojado contiene fallas, se inicia una transferencia de falla en tiempo real, sin que impacte algún servicio o al usuario. Para estos tres niveles, los proveedores establecen ciertos Acuerdos de Nivel de Operación para asegurar sus Acuerdos de Nivel de Servicio (SLA).

Nivel 4: por último, se constituye el nivel de servicio sobre toda la alineación compleja por debajo de donde sólo se puede determinar un impacto negocio del cliente bien fundado. Además, esta determinación será automáticamente en un modelo y algoritmo de sólido en lugar de una interpretación manual.

Nivel 5: muestra la capa de cadenas de servicio que es un nivel importante para los clientes si, por ejemplo, el negocio se distribuye a través de la nube y de diferentes proveedores de servicios.

Desde el punto de vista de un usuario, puede que no haya mucha diferencia en la funcionalidad entre el software tradicional local y una funcionalidad similar en la nube. Las aplicaciones en la nube funcionan en el ecosistema anterior y su funcionalidad debe diseñarse y construirse según el caso, especialmente en los niveles por debajo del nivel superior. Muchos requerimientos funcionales en los modelos de cómputo tradicionales necesitan examinarse y modificarse para su implementación en una nube. Por lo tanto, los requerimientos funcionales del usuario del sistema en la nube pueden ser caracterizados, de manera general, como sigue (Schmietendorf, 2013):

Nivel 1: Los servicios físicos constituyen los procesos funcionales de la arquitectura SOA, incluyendo sus restricciones de Impacto de Negocios (solicitud de (des)abastecimiento de

infraestructura, retiro de Máquinas Virtuales, almacenamiento, ancho de banda de red/ES, CPU, memoria).

Nivel 2: Los servicios agrupados constituyen los procesos funcionales de la arquitectura SOA, incluyendo sus restricciones de Impacto de Negocios (solicitud y (des)abastecimiento de servicio (CRUD (Crear, Leer, Actualizar, Eliminar) y las solicitudes de incidentes, problemas y cambios)

Nivel 3: Los servicios alojados que construyen los procesos funcionales por la arquitectura SOA, incluyendo las restricciones del Acuerdo de Nivel de Operación para asegurar sus Acuerdos de Nivel de Servicio (SLA)

Nivel 4: Los servicios proporcionados constituyen los procesos funcionales de la arquitectura SOA, incluyendo sus restricciones de Impacto de Negocios (Sarbanes-Oxley Audit & Log Requests)

Nivel 5: Los servicios requeridos son los procesos funcionales por parte de los usuarios, incluyendo sus restricciones SLAs de usabilidad y tipo de pago (Live, RealTime Information de KPIs de cumplimiento).

La siguiente figura describe el modelo de software COSMIC genérico para la Arquitectura de Sistemas en la Nube propuesta mediante los patrones COSMIC correspondientes.

Por ejemplo, las aplicaciones en la nube van desde simples aplicaciones personales, a las que pueden tener acceso las aplicaciones móviles, hasta aplicaciones de clase empresarial muy complejas con usuarios en todo el mundo. Por lo regular, las aplicaciones a combinan datos de aplicaciones de negocios y en tiempo real generados por Internet de personas, cosas y máquinas, lo que requiere métodos de ingeniería para el alcance y el modelo de los datos funcionales entre dominios y dispositivos de una manera estandarizada.

Muchos tipos de aplicaciones en la nube evolucionan muy rápidamente con el descubrimiento continuo de nuevos casos de uso basados en patrones de uso/retroalimentación por parte de usuarios funcionales o, a menudo, a través de medios sociales en todo el mundo. Esto requiere de métodos científicos que puedan mapear rápidamente el alcance y medir los cambios de una manera consistente y práctica que sea aceptable y comprensible para todos los interesados, desde los súper usuarios del cliente y los responsables de la administración hasta el personal técnico de ingeniería de software. Aquí es donde un método de medición de tamaño funcional como el método COSMIC puede desempeñar un papel vital.

En la figura 12 se muestra una típica aplicación en la nube y su ecosistema SOA en Internet. Los movimientos de datos COSMIC E-Entry, R-Read, W-Write y X-eXit que contribuyen a "FUR" en diferentes capas e interfaces son claramente identificables usando la estrategia de medición y el modelo de software genérico COSMIC. Es importante reconocer que estos "movimientos de datos" son de naturaleza funcional y visibles para el usuario funcional en ese ámbito de dimensionamiento.

Contexto en la arquitectura en la nube

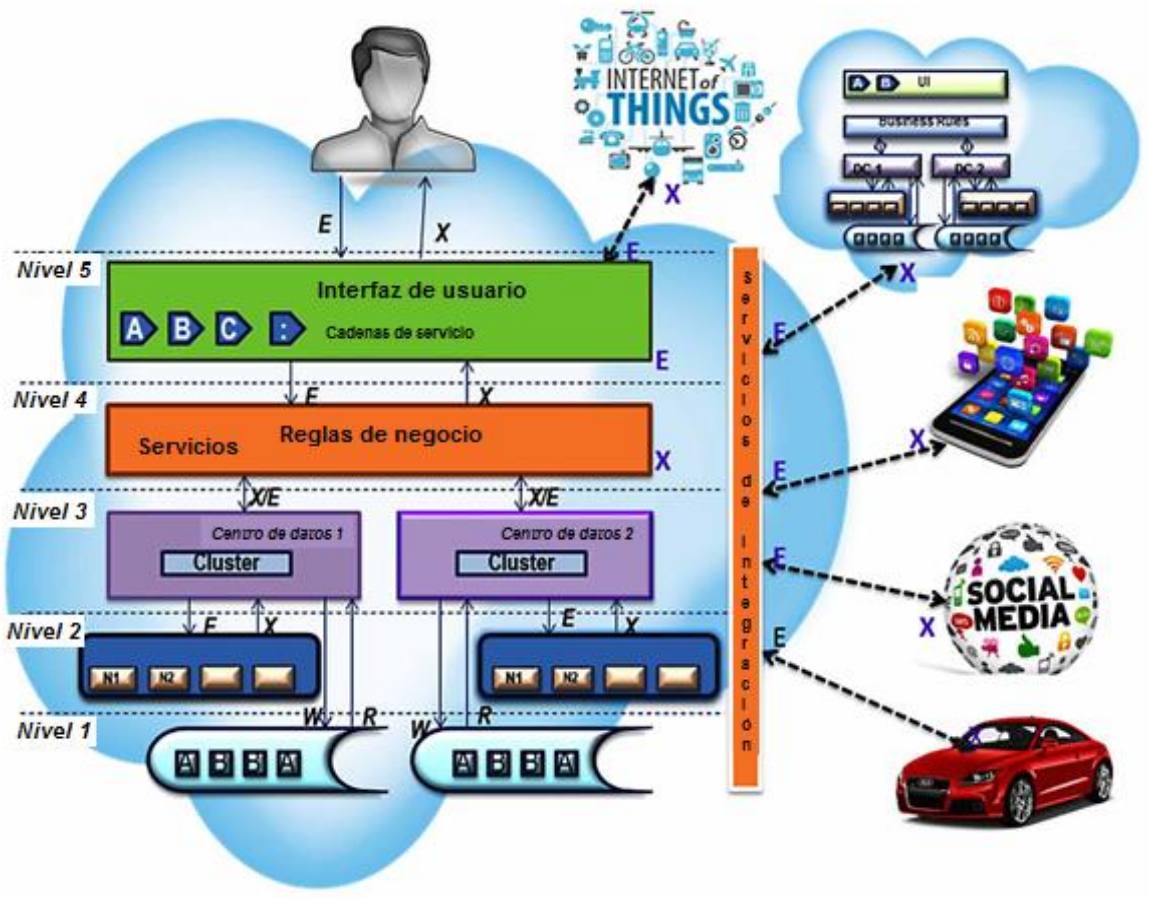
Las aplicaciones en la nube utilizan arquitecturas orientadas a servicios distribuidas con requerimientos funcionales que atraviesan una o más fronteras de la aplicación para usar servicios web de maneras específicas requeridas por diferentes usuarios/aplicaciones. Por lo general, la arquitectura implica múltiples componentes que se comunican entre sí a través de

un mecanismo de acoplamiento suelto tal como una cola de mensajes. Debido a la disposición elástica, la arquitectura debe contener inteligencia en los mecanismos de acoplamiento. También utilizan diferentes formatos de intercambio de datos a través de diferentes capas de la solución. Los componentes de software procesan diferentes tipos de datos y entidades de datos en diferentes capas. El arrendamiento múltiple aporta requerimientos de software que son exclusivos de las soluciones en la nube.

A menudo, los equipos de desarrollo de aplicaciones en la nube constan de múltiples sub-equipos distribuidos que se centran en una capa, un componente o un servicio específico. La productividad de diferentes equipos que trabajan en diferentes capas/componentes utilizando diferentes tecnologías variará y, por lo tanto, se requerirá de una estimación de los esfuerzos en cualquier capa/nivel de componentes. Así es posible estimar de forma rápida y confiable las pequeñas o grandes mejoras a través de la arquitectura orientada a servicios distribuida.

.

Figura 12. Contexto de aplicaciones en la nube



Aplicación de COSMIC en los contextos de uso en la nube

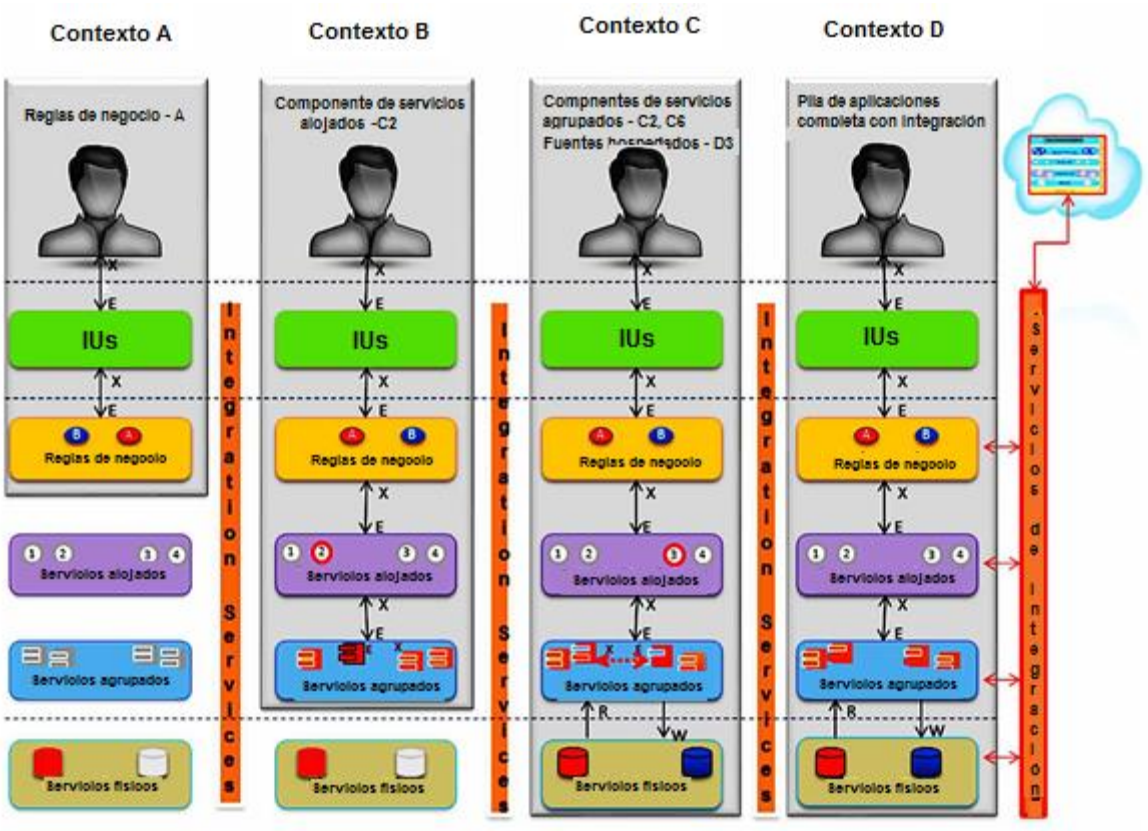
Como se muestra en la figura 13, se pueden observar haber varios escenarios para aplicaciones en la nube y sus correspondientes movimientos de datos pueden medirse mediante COSMIC.

Contexto A: UR en el alcance del proceso de negocios A. Esta interacción en particular no requiere ninguna funcionalidad de ninguna otra capa o componente. Aunque puede tener su mini almacén de datos con reglas de flujo de trabajo, parámetros y configuraciones.

Contexto B: FUR generado a nivel de usuario de la aplicación implica la implementación de cambios funcionales en el servicio alojado "2" y el servicio agrupado CS2 Esto permite a la persona que mide determinar de forma clara la instancia de tamaño aplicable a sus cambios funcionales en su nivel en la pila de nubes, y dibujar la frontera de interacción para identificar eventos activadores y usuarios funcionales en diferentes niveles.

Contexto C: FUR incluye movimientos de datos entre componentes entre CS2 y CS3, así como su correspondiente proceso de negocio B, servicio alojado 3 y componente de datos D5. Esto muestra a la aplicación de COSMIC como movimientos horizontales entre componentes (en dos nodos) dentro del mismo grupo (Cluster).

Figura 13. Ejemplo de movimientos de datos en software en la nube



CS. Estos componentes de servicios de grupos pueden ser desarrollados por diferentes equipos que utilizan diferentes tecnologías al mismo nivel de abstracción.

Contexto D: FUR involucra al usuario utilizando una aplicación de nube externa a través del componente de servicio de integración I6 y los movimientos de datos correspondientes de la pila de aplicaciones completa en donde cada capa interactúa de forma independiente con la capa de integración. Esto muestra a la aplicación de COSMIC en varios escenarios de integración de aplicaciones de negocios híbridas en la nube.

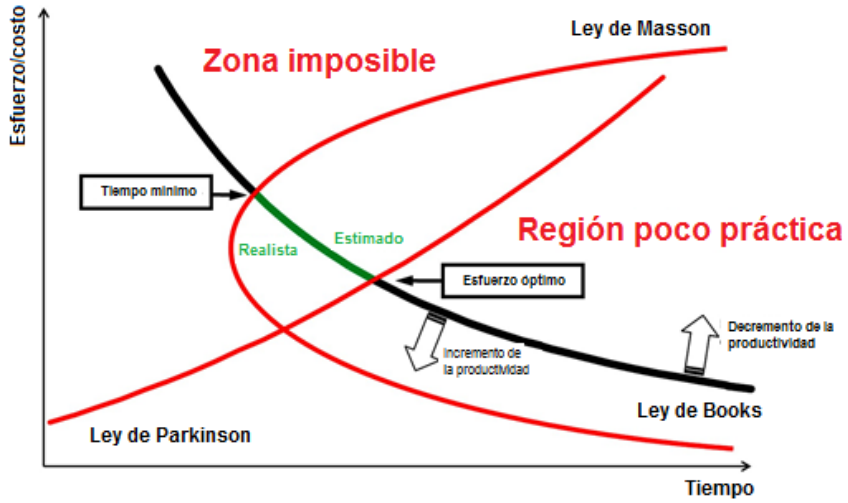
ESTIMACIÓN DEL COSTO, ESFUERZO Y CALENDARIO

Como ya se mencionó, la estimación de costo, esfuerzo y programación es un aspecto muy importante en los proyectos de desarrollo de software comercial, ya que se utiliza para asignar recursos, planificar las nuevas versiones de los productos y negociar pagos entre proveedores y contratistas. En los proyectos de desarrollo de software, el esfuerzo es el parámetro más importante para determinar el costo y el calendario. Por lo tanto, la estimación del costo, esfuerzo y calendario están estrechamente vinculados. El parámetro de entrada más significativo para estimar el esfuerzo y, por lo tanto, el costo y el calendario, es el tamaño del software que se va a desarrollar.

La relación entre tamaño y esfuerzo es la productividad. La productividad es el número de unidades de tamaño que se pueden producir por unidad de tiempo, por ejemplo, CFP/ persona día. La productividad depende de una serie de factores, como el lenguaje de programación, la plataforma de desarrollo, la experiencia del equipo y la madurez de la organización. Los factores que determinan o influyen en la productividad han sido ampliamente estudiados y han llevado a la familia de modelos COCOMO a estimar la productividad en diversas áreas de la ingeniería de software (Boehm, 1981). La mayoría de las herramientas de estimación actuales usan a algunos descendientes de la familia COCOMO. A veces es más conveniente utilizar la inversa de la productividad, la tasa de entrega del producto para reportar cifras de productividad. La unidad más común para la tasa de entrega del producto es hr/CFP. Tenga en cuenta que un mayor número de horas por punto de función COSMIC significa peor productividad. Para una productividad dada, la relación entre tamaño, esfuerzo y horario se rige por la ley de Brooks (Brooks, 1995). Brooks fue el primero en describir que la productividad no era una cifra fija, sino en realidad una función de la presión del tiempo. Por lo tanto, la ley de Brooks también se conoce como la compensación del tiempo/esfuerzo.

En la figura 14, esta compensación se representa desde la parte superior izquierda, hasta la parte inferior derecha (Ross, 2005). Cada punto en la línea representa una situación en la que todos los factores que determinan e influyen en la productividad son los mismos, excepto el calendario requerido (Putnam y Myers, 2003). Cuando la productividad aumenta, la compensación se desplazará hacia la esquina inferior izquierda. Cuando la productividad disminuye, la compensación se desplazará hacia la esquina superior derecha. La ley de Brooks explica que los proyectos que se someten a una presión de tiempo (moviéndose a la izquierda en la figura) requieren más esfuerzo, y por lo tanto cuestan más. Este efecto es causado por el hecho de que las actividades que idealmente son ejecutadas en orden por la misma persona ahora deben ser hechas por diferentes personas en paralelo. Esto aumenta la cantidad de personas necesarias para el personal del proyecto y aumenta el tiempo para coordinar las actividades paralelas (Galorath & Evans, 2006).

Figura 14. Relaciones que gobiernan la estimación paramétrica del costo, esfuerzo y calendario



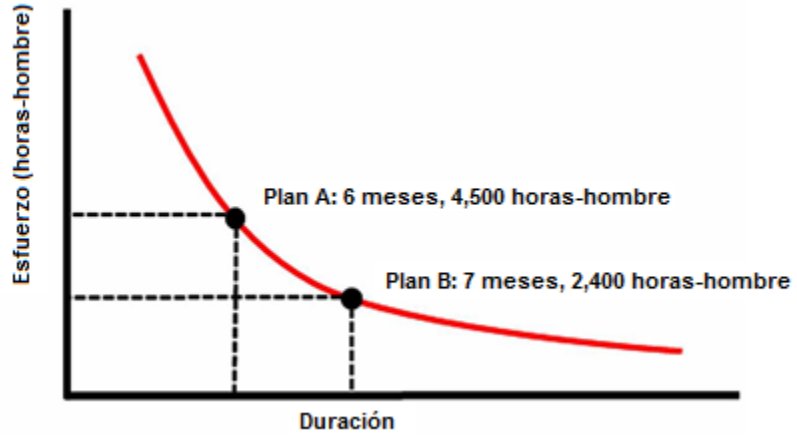
Brooks no definió fronteras a este efecto. En teoría eso significaría que cuando podamos esperar el tiempo suficiente, los proyectos no requerirían casi ningún esfuerzo. A la inversa, cuando tenemos suficientes recursos, los proyectos podrían ser terminados en un día. Paul Masson desarrolló una función que determinó para cada proyecto el tiempo mínimo en el que puede ser terminado (Jensen, 2006). El punto donde la función de Masson se cruza con la ley de Brooks es el tiempo mínimo en el que un proyecto de un tamaño dado y con una configuración de productividad dada puede llevarse a cabo. El área fuera de la función Masson es conocida como la zona imposible. Esto es consistente con el análisis llevado a cabo en 6.300 proyectos hace unos años, donde no se encontró ningún proyecto que fuera terminado antes del tiempo mínimo de desarrollo (Putnam y Myers, 2003). Finalizar el proyecto antes de esa intersección sólo es posible cuando se puede lograr una mayor productividad o cuando el alcance del proyecto disminuye de forma significativa.

También hay un máximo al calendario económicamente realista para que un proyecto termine. Un proyecto no requerirá menos esfuerzo más allá de un cierto punto. Este punto se rige por la ley de Parkinson. La ley de Parkinson fue establecida por primera vez por Cyril Northcote Parkinson como parte de un ensayo humorístico publicado en The Economist en 1955*: El trabajo se expande para llenar el tiempo disponible para su término*. Unos años más tarde, el ensayo se amplió a un libro completo y la ley entró en el mundo de principios económicos serios (Parkinson, 1957). Crear un calendario más allá de la intersección de la ley de Brooks y la ley de Parkinson no llevará a menos esfuerzo para completar el proyecto. Así que la intersección de estas leyes es el punto donde el proyecto puede ser realizado con la cantidad económicamente óptima de esfuerzo. Cualquier punto entre la intersección de la ley de Brooks con las leyes de Paul Masson y Parkinson se considera una estimación realista.

El efecto de esta compensación fue demostrado en un informe de experiencia en 2011 para un proyecto programado en Java de 500 CFP (van Heeringen, 2011). La productividad para construir esta aplicación puede estar entre 0.11 CFP/hr y 0.20 CFP/hr, dependiendo solo de la duración elegida del proyecto, con todas las demás variables constantes (figura 15).

El tamaño del software en COSMIC FP puede utilizarse para estimar el costo, el esfuerzo y el calendario mediante la construcción de modelos paramétricos basados en las reglas anteriores. La mejor práctica para construir tales modelos paramétricos es construirlos con base en los datos internos de la organización (Abran, 2015). Como punto de partida inmediato, los reportes de referencia con base en COSMIC FP del International Software Benchmarking and Standards Group se pueden utilizar para poblar la primera versión de un modelo de estimación (ISBSG, 2012). Las medidas COSMIC se han utilizado con éxito con modelos de estimación basados en Análisis de Regresión, Estimación por Analogía, COCOMO y la curva de Putnam Norden Rayleigh. Los modelos de estimación con base en el tamaño de COSMIC pueden utilizarse para estimar los esfuerzos para el ciclo de vida completo del desarrollo de software o cualquier actividad importante del ciclo de vida como son la fase de pruebas (Kamala Ramasubramani & Abran, 2013).

Figura 15. Compensación tiempo-esfuerzo para un proyecto Java de 500 CFP



COSMIC también se puede utilizar en proyectos de desarrollo de software utilizando enfoques ágiles (*Agile*) (COSMIC, 2011). En 2014, una investigación doctoral en la industria de la vigilancia remota mostró que la inexactitud de la estimación del esfuerzo para proyectos *Agile* podría reducirse del 58%, con base en la técnica de Planning Poker, al 16,5%, con base en COSMIC (Commeyne, 2014). Resultados similares, aunque más de naturaleza cualitativa, han sido reportados por un agente de riesgo australiano (Dekkers, 2014).

La aplicación más importante del método COSMIC es la administración de costos (de proveedores) para el desarrollo y mantenimiento de software. La organización COSMIC mantiene una lista de usuarios conocidos del método (COSMIC, 2015b). La mayoría de las empresas de esa lista utilizan el método de administración de costos. Sólo unos cuantos de ellos han informado con detalle su éxito en el uso del método COSMIC, por ejemplo, en el sector financiero (Vogelezang & Lesterhuis, 2003) y en la industria automotriz (Stern, 2010 y Oriou, 2014).

Otra aplicación del método COSMIC es la estimación del tamaño del código de software. Esta aplicación es particularmente útil para las aplicaciones móviles incorporadas que deben adecuarse en la memoria ROM en productos con presión de alto costo, como son los sistemas de software en el automóvil y las unidades de control electrónico (Lind & Heldal, 2011). Los

investigadores de la industria automovilística sueca mostraron que es posible estimar el tamaño del código con un 15% de precisión. Esto hace posible implementar el software en el producto de memoria lo más pequeño posible.

DIRECCIONES PARA UNA INVESTIGACIÓN FUTURA

El método COSMIC está consiguiendo un reconocimiento más amplio y su uso para diversos entornos está evolucionando. Los principios básicos originales del método COSMIC han permanecido sin cambios desde que se publicaron por primera vez en 1999. Debido a diversos refinamientos, aclaraciones y adiciones, el método ha avanzado hasta la versión 4.0.1 (COSMIC, 2015a). La estructura de la documentación del método es tal que se emiten guías para ofrecer una orientación especial para temas específicos. La orientación que puede ser necesaria para aplicar el método COSMIC en software móvil y de nube se aborda actualmente en guías sobre arquitectura orientada a servicios, software en tiempo real y desarrollo de software ágil. Actualmente se están desarrollando varios casos de estudios de la comunidad móvil.

Uno de los temas que requiere investigación adicional es la estimación temprana del tamaño. De manera especial, el software para móviles y en la nube están evolucionando rápidamente. Estimar el costo, esfuerzo y calendario es una actividad que se necesita más rápido y más temprano y, por lo tanto, debe ser posible obtener estimaciones de tamaño subyacentes con base en especificaciones globales. Ya se han desarrollado varias técnicas de aproximación y se espera una guía para aplicarlas en 2015. Algunas de estas técnicas necesitan ser probadas en la práctica, algunas no han sido probadas aún en el dominio móvil y de la nube.

Otro tema importante, no sólo en el software móvil y en la nube, sino en el desarrollo de software y mantenimiento en general, es el impacto de los requerimientos no funcionales. La experiencia del departamento de Decisión Impulsada por Eventos y Plataformas Móviles Inteligentes (Event-driven Decision & Smart Mobile Paltforms) de IBM ha demostrado que los requerimientos no funcionales de los servicios espacio-temporales representan más del 50% del esfuerzo para producir servicios (IBM Research, 2014). A finales de 2015, una iniciativa conjunta de COSMIC e IFPUG llevó a una clasificación de diferentes tipos de requerimientos no funcionales y restricciones de proyecto que pueden utilizarse para estimar Costo, Esfuerzo y Calendario (COSMIC&IFPUG,2015).

CONCLUSIÓN

El método COSMIC ha demostrado su aplicabilidad en ambientes móviles y está ganando aceptación para la actividad de estimar el desarrollo de software en ambientes en la nube. El método ha sido aceptado como un estándar Internacional, reconocido como una medida independiente de la tecnología para el tamaño funcional del software. El método COSMIC puede ser utilizado por los gerentes de adquisiciones y contratos para seleccionar un proveedor y controlar el desempeño del proveedor. COSMIC puede ser utilizado por los inversionistas para decidir sobre el análisis costo/beneficio para nuevos proyectos y para valorar activos de software. El método COSMIC es una medida de tamaño independiente que puede ser utilizada por los Gerentes de Proyecto para estimar, programar proyectos y controlar los cambios del alcance en los proyectos. La respuesta de los usuarios de COSMIC revela que los

requerimientos se vuelven inequívocos, rastreables y probables cuando COSMIC se utiliza para el medir el tamaño, lo que resulta en una mayor satisfacción del cliente.

Las principales ventajas del método COSMIC son:

- Aproximación temprana del tamaño, como base para la estimación del costo, esfuerzo y calendario en fases tempranas del ciclo de vida del desarrollo.
- Aplicable en arquitectura distribuida y distribuida.
- Aplicable a diferentes tipos de software (componentes) en diferentes capas arquitectónicas.
- Adecuado para diferentes tipos de aplicaciones (transaccional, en tiempo real, *Data Warehouse*).
- Independiente de la tecnología y plataforma.
- Aplicable para metodologías de desarrollo de software lineales, iterativas y ágiles.
- Estándar abierto, conforme a la meta-norma ISO para medir el tamaño funcional.
- Apoyado por una comunidad mundial de voluntarios.

ACERCA DE COSMIC

COSMIC es una organización voluntaria y sin fines de lucro de expertos en métricas de software, que representa a profesionales de la industria y académicos de más de veinte países de todo el mundo. COSMIC fue fundada en 1998. Todas sus publicaciones están 'abiertas' y están disponibles para distribución gratuita sujetas a restricciones de derechos de autor y reconocimiento.

Los principios del método COSMIC se establecieron en 1999, después de que se llevaron a cabo pruebas de campo durante 2000-2001 con varias empresas internacionales e instituciones académicas antes de anunciar el método. ISO adoptó COSMIC como Norma Internacional y lo anunció como ISO/IEC 19761 en 2003. En la actualidad, el Manual de Medición v4.0.1 sirve como referencia para la medición COSMIC (COSMIC, 2015a). Se han publicado varios documentos adicionales para apoyar el método y pueden descargarse de forma gratuita (COSMIC, 2014).

REFERENCIAS

Abran, A. (2015). *Software Project Estimating: The fundamentals for providing high quality information to decision makers*. Hoboken, NJ: John Wiley & Sons.

Abran, A., Symons, C. R., & Oligny, S. (2001). An overview of COSMIC field trial results. In *Proceedings of the 12th European Software Control and Metrics conference*.

Boehm, B. W. (1981). *Software Engineering Economics*. Upper Saddle River, NJ: Prentice Hall PTR.

Brooks, F. P. Jr. (1995). *The Mythical Man-Month*. Boston, MA: Addison-Wesley Longman Publishing.

Commeyne, C. (2014). *Établissement d'un modèle d'estimation a posteriori de projets de maintenance de logiciels*. (PhD thesis). École de Technologie Supérieure, Montréal, Canada.

Common Software Measurement International Consortium. (2010). *Guideline for Sizing Service Oriented Architecture Software*. Montréal, Canada.

Common Software Measurement International Consortium. (2011). *Guideline for the use of COSMIC FSM to manage Agile projects*. Montréal, Canada.

Common Software Measurement International Consortium. (2012). *Guideline for Sizing Real-time Software*. Montréal, Canada.

Common Software Measurement International Consortium. (2014). *Publications*. Retrieved May 20, 2015, from http://cosmic-sizing.org/publications

Common Software Measurement International Consortium. (2015a). *Measurement Manual (version 4.0.1)*. Montréal, Canada.

Common Software Measurement International Consortium. (2015b). *Usage of the COSMIC method*. Retrieved May 20, 2015, from http://cosmic-sizing.org/cosmic/usage/

Common Software Measurement International Consortium & International Function Point User Group. (2015). *Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating*. Montréal, Canada: Author.

Dekkers, C. (2014). *Function Points and Agile Development... Considerations and Opportunities*. Retrieved May 20, 2015, from https://www.linkedin.com/grp/post/2758144-5904564005854285828

Galorath, D. D., & Evans, M. W. (2006). *Software Sizing, Estimation, and Risk Management*. Boca Raton, FL: Auerbach Publications. doi:10.1201/9781420013122

IBM Research. (2014). *SaS hub non-functional requirements*. Retrieved May 20, 2015, from https:// www.research.ibm.com/haifa/projects/software/nfr/index.html

Institute of Electrical and Electronics Engineers Computer Society. (2014). *Guide to the Software Engineering Body of Knowledge (version 3.0)*. Washington, DC: IEEE.

International Organization for Standardization. (2007). *Information Technology– Software Measurement - Functional Size Measurement. International Standard ISO/IEC 14143:2007*. Geneva, Switzerland: ISO.

International Organization for Standardization. (2011*). Software Engineering – COSMIC – A Functional Size Measurement Method. International Standard ISO/IEC 19761:2011*. Geneva, Switzerland: ISO.

International Software Benchmarking and Standards Group. (2012). *The Performance of Real-time, Business and Component Software Projects – An analysis of COSMIC measured projects in the ISBSG database*. Balaclava, Australia: ISBSG.

Janssen, C. (2014). *Three-Tier Architecture*. Retrieved May 20, 2015, from http://www.techopedia.com/ definition/24649/three-tier-architecture

Jensen, R. W., Putnam, L. H. Sr, & Roetzheim, W. (2006). *Software estimating models: Three viewpoints*. *Crosstalk*, 2, 23–29.

Jones, C. (2013). *A short history of the Lines of Codes (LoC) metric*. Retrieved May 20, 2015, from http://namcookanalytics.com/a-short-history-of-the-lines-of-code-loc-metric

Kamala Ramasubramani, J., & Abran, A. (2013). *A survey of Software Test Estimation Techniques. Journal of Software Engineering and Applications*, 6(10), 47–52. doi:10.4236/jsea.2013.610A006

Lind, K., & Heldal, R. (2011). *A model-based and automated approach to size estimation of embedded software components. In Proceedings of the 14th international conference on Model Driven Engineering Languages and Systems (MODELS)*. Wellington, Nueva Zelanda: Springer Verlag. doi:10.1007/978-3- 642-24485-8_24

National Institute of Standards and Technology. (2011). *The NIST Definition of Cloud Computing*. Retrieved May 20, 2015 from http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Oriou, A., Bronca, E., Bouzid, B., Guetta, O., & Guillard, K. (2014). Manage the automotive embedded software development cost & productivity with the automation of a functional size measurement method (COSMIC). In *Proceedings of the Joint Conference of the 24th International Workshop on Software Measurement (IWSM) and the 9th International Conference on Software Process and Product Measurement (Mensura)*. Rotterdam, The Netherlands: IEEE Computer Society Conference Publishing Services.

Parkinson, C. N. (1957). Parkinson's law, or the pursuit of progress. Cutchogue, NY: Buccaneer Books.

Putnam, L. H., & Myers, W. (2003). *Five Core Metrics: The intelligence behind successful software management*. New York, NY: Dorset House Publishing.

Ross, M. A. (2005). *Parametric project monitoring and control: performance-based progress assessment and prediction*. Retrieved May 20, 2015, from http://www.dtic.mil/ndia/2005cmmi/wednesday/ross3.pdf

Schmietendorf, A., Fiegler, A., Neumann, R., Wille, C., & Dumke, R. R. (2013). COSMIC Functional Size Measurement of Cloud Systems. In *Proceedings of the Joint Conference of the 23rd International Workshop on Software Measurement (IWSM) and the 8th International Conference on Software Process and Product Measurement (Mensura)*. Ankara, Turkey: IEEE Computer Society Conference Publishing Services. doi:10.1109/IWSM-Mensura.2013.15

Stern, S., & Guetta, O. (2010). Manage the automotive embedded software development cost by using a Functional Size Measurement Method (COSMIC). In *Proceedings of the 4thEmbedded Real Time Software and Systems conference*. Toulouse, France: ERTS2.

van Heeringen, H. S. (2011). Estimate faster, cheaper... and better! In *Proceedings of the 8th Software Measurement European Forum*. Rome, Italy.

van Heeringen, H. S., & van Gorp, E. W. M. (2014). Measure the functional size of a mobile app using the COSMIC functional size measurement method. In *Proceedings of the Joint Conference of the 24th International Workshop on Software Measurement (IWSM) and the 9th International Conference on Software Process and Product Measurement (Mensura)*.

Rotterdam, The Netherlands: IEEE Computer Society Conference Publishing Services. doi:10.1109/IWSM.Mensura.2014.8

Vogelezang, F. W. (2013a). *The first generation of Functional Size Measurement*. Retrieved May 20, 2015, from http://thePriceofIT.blogspot.nl/2013/01/the-first-generation-FSM.html

Vogelezang, F. W. (2013b). *What is a second generation FSM method*. Retrieved May 20, 2015 from http://thepriceofit.blogspot.nl/2013/02/second-generation-FSM.html

Vogelezang, F. W., & Lesterhuis, A. (2003). Applicability of COSMIC in an administrative environment - Experiences of an early adopter. In *Proceedings of the 13th International Workshop on Software Measurement*. Montréal, Canada: Shaker Verlag.

LECTURAS ADICIONALES

Abran, A. (2015). *Software Project Estimating: The fundamentals for providing high quality information to decision makers*. Hoboken, NJ: John Wiley & Sons.

Dumke, R. R., & Abran, A. (2011). *COSMIC function points: theory and advanced practices*. Boca Raton, FL: Auerbach Publications.

Dumke, R. R., Schmietendorf, A., Seufert, M., & Wille, C. (2014). *Handbuch der Software Umfangs- messung und Aufwandschätzung*. Berlin, Germany: Logos Verlag.

Symons, C. R., & Lesterhuis, A. (2014). *Introduction to the COSMIC method of measuring software*. Montréal, Canada: COSMIC.

TÉRMINOS CLAVE Y DEFINICIONES

Movimiento de datos: Un componente funcional base que mueve un solo tipo de grupo de datos. Existen cuatro tipos de movimientos de datos: entrada, salida, lectura y escritura (COSMIC, 2015a)

Entrada: Una entrada es un movimiento de datos que mueve a un grupo de datos desde el usuario funcional a través de la frontera al proceso funcional donde es requerido. Se considera que una entrada también incluye cierto manejo de datos asociados (validación).

Salida: Una salida es un movimiento de datos que mueve a un grupo de datos desde un proceso funcional a través de la frontera al usuario funcional donde es requerido. Se considera que una salida también incluye cierto manejo de datos asociados (validación).

Proceso funcional: Un conjunto de movimientos de datos que representan una parte elemental de los requerimientos funcionales del usuario (FUR) para el software que se está midiendo, que es único dentro de los requerimientos funcionales del usuario y que puede definirse de manera independiente de cualquier otro proceso funcional en los requerimientos funcionales del usuario. Un proceso funcional puede tener solo una sola entrada activadora. Cada proceso funcional inicia el procesamiento al recibir un grupo de datos movido por la entrada activadora del proceso funcional. El conjunto de todos los movimientos de datos de un proceso funcional es el conjunto que se necesita para satisfacer los requerimientos funcionales del usuario para todas las posibles respuestas a esa entrada activadora.

Requerimientos funcionales de usuario (FUR por sus siglas en inglés**):** Un subconjunto de requerimientos del usuario. Son los requerimientos que pueden describir qué debe hacer el software en términos de tareas y servicios. Los requerimientos funcionales del usuario se refieren, pero no se limitan a: transferencia de datos (por ejemplo, introducir datos del cliente, enviar señales de control, etc.) transformación de datos (por ejemplo, calcular un interés bancario, deducir la temperatura promedio, etc.) almacenar datos (por ejemplo, almacenar una orden del cliente, registrar una temperatura ambiente en el transcurso del tiempo, etc.) recuperar datos (por ejemplo, listar los empleados actuales, recuperar la última posición de una aeronave, etc.). Ejemplos de requerimientos de usuario que no son requerimientos funcionales incluyen, pero no se limitan a, restricciones en la calidad (por ejemplo, usabilidad, confiabilidad, eficiencia y portabilidad), restricciones organizacionales (por ejemplo, hardware y cumplimiento a estándares), restricciones ambientales (por ejemplo, interoperabilidad, seguridad, privacidad y protección), restricciones de implementación (por ejemplo, lenguaje de desarrollo, fechas de entrega).

Capa: Una partición funcional de una arquitectura de sistema software. El software es una capa que proporciona un conjunto de servicios que es cohesivo de acuerdo con criterios definidos y, el software en otras capas, puede utilizarse sin saber cómo se implementan esos servicios. La relación entre el software en cualquiera de las dos capas está definida como una "regla de correspondencia" que puede ser "jerárquica", es decir, el software en la capa A se le permite utilizar los servicios proporcionados por el software de la capa B, pero no viceversa (en donde la relación jerárquica puede ser hacia arriba, hacia abajo o lateral), o "bidireccional", es decir, el software en la capa A se le permite utilizar el software de la capa B y viceversa. El software en una capa intercambia grupos de datos con software en otra capa mediante sus procesos funcionales respectivos. El software en una capa no necesariamente utiliza todos los servicios funcionales suministrados por el software en otra capa. Una capa puede ser dividida en otras capas de acuerdo con las diferentes arquitecturas de software definidas (COSMIC, 2015a)

Requerimientos no funcionales: Cualquier requerimiento o restricción para un sistema hardware/software o un producto de software, o para un proyecto a desarrollar o mantener como un sistema o producto, excepto un requerimiento funcional de usuario para el software. Considere que los requerimientos del sistema o software que son expresados al inicio como no funcionales, a menudo evolucionan a medida que un proyecto avanza total o parcialmente en los requerimientos funcionales del usuario para el software (COSMIC, 2015a).

Almacenamiento persistente: Almacenamiento que permite a un proceso funcional almacenar un grupo de datos más allá de la vida del proceso funcional y/o del que 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 el mismo proceso COSMIC, 2015a).

Lectura: Una lectura es un movimiento de datos que mueve un grupo de datos del almacenamiento persistente al proceso funcional que lo requiere Se considera que una lectura incluye cierto manejo de datos asociado.

Evento activador o desencadenante: Un evento reconocido en los requerimientos funcionales del usuario que está siendo medido, que hace que uno o más usuarios funcionales del software generen uno o más grupos de datos, cada uno de los cuales será movido posteriormente por una entrada activadora. Un evento activador no se puede subdividir y ha

ocurrido o no ha ocurrido. Considere que los eventos de reloj y de tiempo pueden ser eventos activadores (COSMIC, 2015a).

Usuario (Usuario funcional): Un (tipo de) usuario que es un remitente y/o destinatario de datos en los requerimientos funcionales del usuario de una pieza de software. Los usuarios funcionales de la parte de software a medir se establecen de acuerdo al propósito de la medición. Cuando el propósito de medir una parte de software está relacionado con el esfuerzo por desarrollar o modificar la pieza de software, entonces los usuarios funcionales deben ser todos los remitentes y/o destinatarios de datos de/hacia la funcionalidad nueva o modificada, como se definen en los requerimientos funcionales del usuario (COSMIC, 2015a).

Escritura: Una escritura es un movimiento de datos que mueve un grupo de datos que se encuentra dentro de un proceso funcional para un almacenamiento persistente. Se considera que una escritura incluye cierta manipulación de datos asociada.