ISSN 1409-2441

Ingeniería

Revista Semestral de la Universidad de Costa Rica

Vol. 16 (2) Ago/Dic 2006

CONTENIDO

Artículos

Nota técnica

Normas

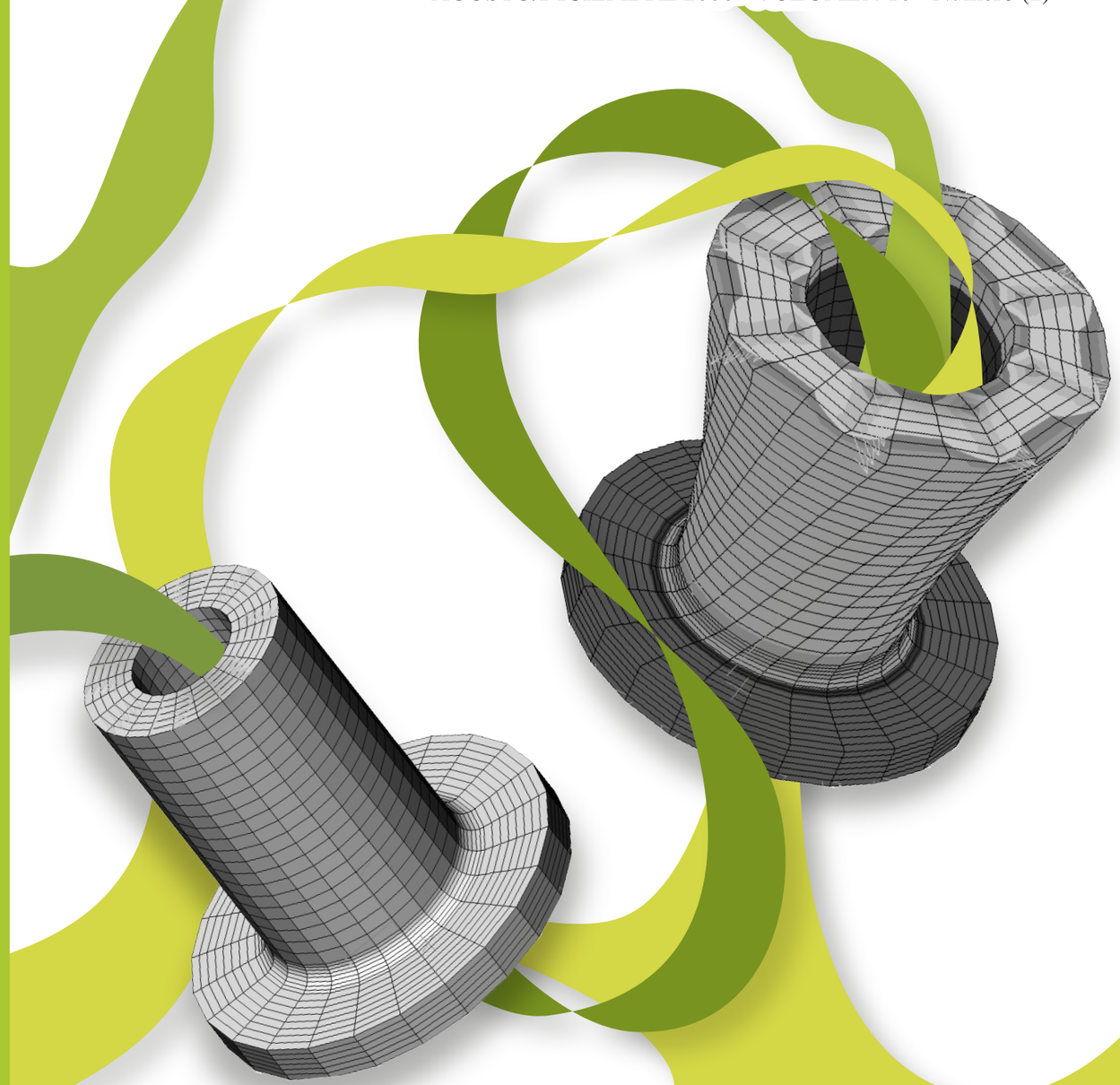
EDITORIAL DE LA UNIVERSIDAD DE COSTA RICA

http://www.revistaingenieria.inii.ucr.ac.cr
www.ucr.ac.cr
ISSN 1409-2441

Ingeniería

Revista de la Universidad de Costa Rica

AGOSTO/DICIEMBRE 2006 - VOLUMEN 16 - Número (2)



Número (2)
Agosto / Diciembre 2006
Volumen 16
REVISTA INGENIERÍA

Nota técnica

Ingeniería 16 (2): 115-127, ISSN: 1409-2441; 2006. San José, Costa Rica

EXPERIENCIA DOCENTE EN LA UNIVERSIDAD DE COSTA RICA EN EL USO DE PUNTOS DE FUNCIÓN Y METODOLOGÍAS ORIENTADAS A OBJETOS PARA ESTIMAR PROYECTOS DE SOFTWARE

Gabriela Salazar Bermúdez

Resumen

La estimación de proyectos de software es uno de los temas actuales más importantes dentro de la disciplina de la ingeniería de software. Este artículo describe la experiencia al enseñar a estimar y planificar un proyecto de software a estudiantes de pregrado de la carrera de Ciencias de la Computación e Informática en la Universidad de Costa Rica. Se describe en detalle la metodología para estimar un proyecto de software, en una etapa en donde apenas se está conceptualizando, utilizando puntos de función y metodologías orientadas a objetos. Además, se explica cómo derivar el esfuerzo requerido para implementarlo con base en el tamaño estimado. Los puntos descritos en este artículo pueden interesar a profesores e instructores que deseen formar a futuros ingenieros de software en el campo de la planificación.

Palabras clave: estimación, métricas, tamaño, esfuerzo, metodologías orientadas a objetos.

Abstract

Right now software project estimation is one of the most important subjects of the software engineering discipline. This article describes the teaching experience on estimation and planning of software projects, to bachelor's and master's degree students programs on Computer Science and Informatics in the University of Costa Rica. It describes in detail: the methodology to estimate and plan, object oriented methodologies to conceptualize the software to be developed, and Function Points to measure its size. It is also explained, how to derive effort and duration based on size estimation. These described items could interest teachers and instructors who wish to form future software engineers on planning fields.

Key words: estimation, measures, size, effort, object oriented methods.

Recibido: 07 de julio del 2006 • **Aprobado:** 18 de enero del 2007

1. INTRODUCCIÓN

En la mayoría de las empresas donde se produce software para apoyar el negocio, las prácticas de estimación y planificación son débiles. En general, los administradores estiman el costo y la duración del proyecto por desarrollar, utilizando solamente el juicio de un experto. Con el fin de apoyar a la industria desarrolladora de software en Costa Rica, la Universidad de Costa Rica, como motor impulsor de las últimas prácticas internacionales, ha considerado la iniciativa de enseñarles a sus estudiantes, las nuevas metodologías de estimación y planificación de proyectos de software. Por esta razón, aproximadamente desde 1996, se les ha enseñado a estimar el tamaño de la aplicación utilizando los Puntos de Función (PF) y con base en el tamaño, a estimar el esfuerzo requerido para desarrollar la aplicación.

El Análisis de Puntos de Función (en inglés, Function Point Analysis, FPA) fue introducido por Albrecht en 1970. Su propósito era solucionar algunos de los problemas asociados con el cálculo del tamaño del software en Líneas de Código (en inglés "Lines of code", LOC) y las medidas de productividad que se daban, especialmente por las diferencias en los cálculos de LOC que resultaban de los diferentes niveles de lenguajes que se utilizaban. A él le interesaba medir la funcionalidad del software desde el punto de vista del usuario, independientemente de su implementación, por lo que introdujo los PF como una medida del tamaño de una aplicación, desde el punto de vista funcional o del usuario. Los PF son derivados de aspectos externos de las aplicaciones de software, los cuales son: entradas, salidas, consultas, archivos lógicos e interfaces (Kan, 2003).

Desde entonces, el uso de los PF ha ganado aceptación como una medida de productividad clave y los procedimientos de conteo han sido actualizados varias veces desde su primera publicación. El FPA es ahora administrado por el International Function Point Users Group (IFPUG). Ellos proveen los estándares para aplicar el cálculo de los PF a través de su publicación "Counting Practices Manuals" (Garmus, 2001).

2. ANTECEDENTES

Después de haber enseñado las prácticas recomendadas del FPA durante aproximadamente cuatro años, se detectaron fundamentalmente dos problemas:

- En primer lugar, esta técnica requiere disponer de información muy detallada en una fase temprana, en donde apenas se está evaluando la viabilidad del proyecto. Las reglas de conteo del FPA se deben aplicar a dos elementos que son: los archivos y las transacciones de la aplicación que se está estimando. Esta reglas requieren conocer cierto nivel de detalle de dichos elementos, para poder determinar su complejidad y posteriormente, el tamaño de la aplicación.

- En segundo lugar, para identificar los elementos mencionados en el párrafo anterior y realizar la estimación, se ha utilizado el diagrama de entidad-relación, el cual es típicamente una herramienta de las técnicas de diseño estructurado. Esto ha provocado controversia, porque los estudiantes utilizan métodos orientados a objetos para desarrollar, y los modelos creados con estos métodos, son diferentes especialmente en las fases tempranas, debido a que no proveen documentación tradicional como: diagramas de entidad-relación, estructuras de bases de datos o modelos de procesos jerárquicos, sino que modelan el sistema como un conjunto de objetos cooperando.

Este artículo propone una metodología para estimar el tamaño de una aplicación, utilizando los métodos orientados a objetos y los procedimientos de FPA, pero intentando corregir los problemas mencionados anteriormente. En la sección tres se explican algunos conceptos teóricos para entender la metodología propuesta, la cual se describe en la sección 4 de este documento. Finalmente, en la sección 5 se presentan las conclusiones.

3. MARCO TEÓRICO

3.1. Modelo de casos de uso

El modelo principal del Object-Oriented Software Engineering (OOSE) propuesto por Jacobson, es el modelo de casos de uso y es la base sobre la cual se desarrollan los otros modelos como: el de objetos del dominio, el de análisis, el de clases, el de implementación y el de pruebas. Este modelo tiene dos elementos: los casos de uso y los actores.

De acuerdo con Larman, (1999) "Un caso de uso describe un proceso, un proceso de negocios por ejemplo. Un proceso describe, de comienzo a fin una secuencia de los eventos, de las acciones y las transacciones que se requieren para producir u obtener algo de valor para una empresa o actor. Algunos procesos pueden ser: solicitar un pedido, retirar efectivo de un cajero automático".

Larman recomienda que durante las fases de conceptualización y planeación, después de haber identificado las funciones del sistema y definido sus fronteras se debe hacer lo siguiente: describir los casos de uso en formato esencial y dibujarlos en un diagrama de casos de uso, incluyendo las relaciones entre ellos. A continuación se muestra un ejemplo de la descripción esencial del caso de uso "Comprar productos" y en la Figura 1 se muestra el diagrama de casos de uso.

Caso de uso: Comprar productos

Actores: Cliente, Cajero

Descripción: Un cliente llega a una caja con productos que desea comprar. El cajero registra los productos y obtiene el pago de parte del cliente. Al terminar la transacción, el cliente se marcha con los productos.

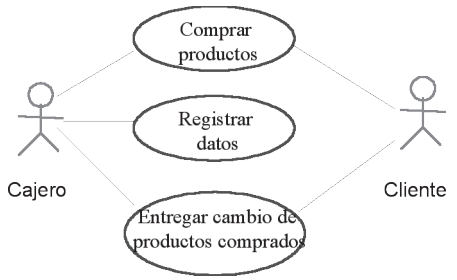


Figura 1. Ejemplo de un diagrama de casos de uso.
Fuente: (Larman,1999)

Los casos de uso pueden ser refinados con las relaciones "include" y "extend". En una relación de inclusión ("include"), las partes comunes pueden ser extraídas y modeladas para ser "usadas" por otros casos de uso. En una relación de "extensión" ("extend") un caso de uso puede ser insertado en un flujo de acción de un caso de uso existente, la extensión agrega funcionalidad, independientemente del existente (Larman, 1999).

Posteriormente, conforme se avance en la investigación de los requerimientos y se comience a entender mejor el alcance del problema y los requerimientos, los casos de uso se describirán en un formato expandido, el cual, a diferencia del formato esencial agrega una sección para el curso normal de los eventos, en donde se describe paso a paso cada evento y otra para las excepciones (Larman, 1999).

3.2 Modelo de objetos del dominio

Es una identificación y representación de los objetos relevantes al dominio del problema. Un objeto es un concepto, abstracción, o cosa con fronteras bien definidas y con significado para una aplicación. Larman (1999) recomienda crear este modelo durante la fase de planeación, para comprender los aspectos generales del sistema, entender el glosario y efectuar las estimaciones. Él indica que: "no debe hacerse una investigación exhaustiva para no saturar la investigación desde el principio y evitar exceso de complejidad". En el enfoque de Jacobson este modelo es opcional durante el análisis de requerimientos.

3.3 Modelo de análisis

El modelo de análisis es una representación de objetos clasificada en tres tipos: entidad, interfaces y control (Figura 2). Este modelo es derivado del modelo de casos de uso. Toma la funcionalidad de cada caso de uso, la particiona en objetos y a cada objeto le asigna una clasificación. El objetivo de esta clasificación es crear una estructura adaptable a los cambios. De manera que, si por ejemplo cambian los requerimientos de interfaz, solo se tengan que cambiar los objetos de interfaz.

Los objetos de entidad modelan la información que va a existir en el sistema en forma persistente, reflejan abstracciones de entidades del mundo real. Los objetos de interfaz se usan para modelar las interfaces del sistema. Proveen la interfaz del sistema al usuario o a otro sistema (es decir, la interfaz a un actor). Los objetos de control modelan la secuencia específica de uno o más casos de uso. Coordinan los eventos para realizar la conducta especificada en el caso de uso (Quatrani, 2003).

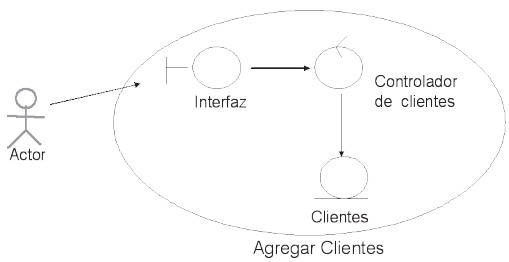


Figura 2. Modelo de análisis.
Fuente: (Quatrani, 2003)

3.4 Técnica de análisis puntos de función

La metodología de FPA explicada en Garmus, (2001) puede resumirse en los siguientes siete pasos:

Paso 1. Determinar el tipo de conteo de puntos de función. Se determina el tipo de conteo de acuerdo con tres posibilidades: para proyectos en desarrollo, para mejora de proyectos y para aplicaciones ya desarrolladas.

Paso 2. Identificar el alcance y las fronteras de la aplicación que se está estimando. La frontera es el límite entre el proyecto o aplicación que está siendo medida y las aplicaciones externas o el dominio del usuario. En la Figura 3 se muestra la funcionalidad reconocida para el conteo. Se puede observar cómo los usuarios y las aplicaciones externas pueden interactuar con la aplicación a través de las entradas externas, consultas externas y salidas externas.

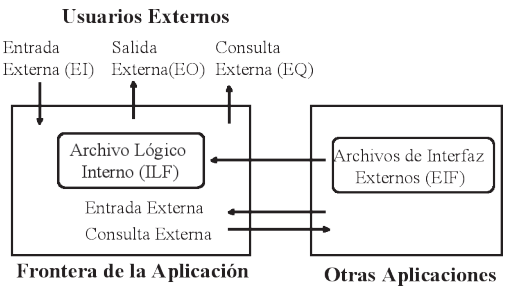


Figura 3. Funcionalidad reconocida en el conteo de Puntos de Función.

Fuente: (Garmus, 2001)

Paso 3. Identificar todas las funciones de datos y su complejidad. Las funciones de datos se clasifican en Archivos Lógicos Internos (en inglés Internal Logical Files, ILF) y Archivos de Interfaz Externos (en inglés External Interface Files, EIF) y se explican a continuación:

- **Archivos Lógicos Internos (en adelante ILF):** Es un grupo de datos relacionados lógicamente e identificables por el usuario, que se almacena completamente dentro de las fronteras del sistema y sus datos son administrados a través de uno o más procesos elementales (es decir, a través de Entradas Externas que se explicarán más adelante).

- **Archivos de Interfaz Externos (en adelante EIF):** Es un grupo de datos relacionados lógicamente e identificables por el usuario, que se utilizan solamente para fines de referencia por la aplicación que se está estimando, ya que los datos son administrados por las Entradas Externas de otras aplicaciones, es decir, cada EIF es un ILF de otra aplicación.

El conteo físico de ILF y EIF junto con la complejidad relativa de cada uno, determina la contribución a los PF desajustados. Esta complejidad se encuentra determinada por el número de elementos de datos o atributos del archivo (en inglés "data element types", DET) y de tipos de registros (en inglés "record element types", RET) asociados a cada uno. Los RET son los grupos o subgrupos que constituyen una parte de un archivo; en el enfoque OOSE se puede ver como una agregación de un objeto. Dependiendo del número de DET y RET de los ILF y EIF la metodología FPA propone determinados pesos, los cuales ayudan a establecer si la complejidad es alta, media o baja. Para más referencia consultar (Garmus, 2001).

Paso 4. Identificar todas las funciones transaccionales y su complejidad. Las funciones transaccionales se clasifican en Entradas Externas, Salidas Externas y Consultas Externas y se describen de la siguiente manera:

- **Entradas Externas (EI, External Inputs en inglés):** Es un proceso elemental que procesa datos o información de control que ingresan a la aplicación. Los datos procesados administran uno o más ILF. La información de control puede o no administrar un ILF. La intención de un EI es administrar uno o más ILF y alterar la conducta de la aplicación a través del procesamiento lógico. La tarea de administrar puede ser reconocida por los verbos: agregar, modificar, eliminar, actualizar, asignar, evaluar, guardar como y crear.

- **Salidas Externas (EO, External Outputs en inglés):** Es un proceso elemental que genera datos o información de control desde adentro hacia afuera de la aplicación. El objetivo principal es presentar información al usuario mediante el procesamiento lógico, el cual debe contener al menos una fórmula

matemática, cálculo o crear datos derivados, los cuales se calculan a partir de otros. Adicionalmente, un EO puede actualizar uno o más ILF y/o alterar el comportamiento del sistema.

- **Consultas Externas (EQ, External Inquiries en inglés):** Es un proceso elemental que recupera datos o información de control para enviarlos fuera de la aplicación. El objetivo principal es mostrar información al usuario a través de la recuperación de datos o información de control desde un ILF o un EIF. El procesamiento lógico no contiene fórmulas matemáticas o cálculos, ni crea datos derivados. Ningún ILF es administrado durante el procesamiento por un EQ, por lo que la conducta de la aplicación no se altera.

El conteo físico de EI, EO y EQ, junto con la complejidad relativa de cada uno, determina la contribución a los PF desajustados. Esta complejidad se encuentra determinada por el número de elementos de datos (o DET) y de archivos referenciados (en inglés "file types referenced", FTR) asociados a cada transacción. Los FTR para las transacciones EI y EO corresponden al número total de ILF administrados o leídos y de EIF leídos por dichas transacciones. En el caso de los EQ los FTR se refieren al número total de ILF y EIF leídos por las transacciones EQ.

Dependiendo del número de DET y de FTR en las transacciones EI, EO y EQ, la metodología FPA propone determinados pesos los cuales ayudan a establecer si la complejidad es alta, media o baja Para más referencia consultar Garmus, (2001).

Paso 5. Determinar los puntos de función sin ajustar. Para obtener el total de Puntos de Función Sin Ajustar (en adelante PFSA) se utilizan la información obtenida en los pasos anteriores y, dependiendo de la complejidad de cada archivo o transacción, se le asigna un peso de acuerdo con el Cuadro 1.

Paso 6. Determinar el valor del Factor de Ajuste. Con el fin de adaptar la estimación de los PFSA a las condiciones de trabajo bajo las cuales la aplicación va a ser desarrollada, se debe calcular el Factor de Ajuste (FA). Para estimar el FA se debe aplicar la fórmula (1), pero antes se debe estimar el Grado Total de Influencia (GTI).

$$FA = (GTI \bullet 0{,}1) + 0{,}65 \qquad (1)$$

El GTI corresponde a la suma de catorce características las cuales son: Comunicación de Datos, Procesamiento de Distribuido de Datos, Rendimiento, Configuración fuertemente utilizada, Frecuencia de Transacciones, Frecuencia de Transacciones, Entrada de Datos en Línea, Diseño para la eficiencia del Usuario Final, Actualización de datos en Línea, Procesamiento complejo, Reusabilidad del código, Facilidad de instalación, Facilidad de operación (Soporte de respaldo), Instalación en distintos lugares, Facilidad de cambio.

A cada una de esas características se le asigna un peso entre 0 y 5 que indica la importancia de la característica para el sistema bajo análisis. El peso 0 indica que la característica no presenta ninguna

Cuadro 1. Pesos.

	Complejidad		
Componentes	**Baja**	**Media**	**Alta**
Archivos Lógicos Internos (ILF)	x7	x10	x15
Archivos de Interfaz Externos (EIF)	x5	x7	x10
Entradas Externas (EI)	x3	x4	x6
Salidas Externas (EO)	x4	x5	x7
Consultas Externas (EQ)	x3	x4	X6
Total PFSA			

Fuente: (Garmus, 2001)

influencia en el desarrollo de la aplicación y 5 indica que influye fuertemente.

Paso 7. Calcular los puntos de función ajustados. Finalmente los Puntos de Función finales (ya ajustados) se obtienen a través de la fórmula (2) y es el resultado de multiplicar Puntos de Función sin Ajustar por el Factor de Ajuste.

$$PF = PFSA \bullet FA \qquad (2)$$

4. METODOLOGÍA PROPUESTA PARA ESTIMAR Y PLANIFICAR

A continuación se explica la metodología propuesta para estimar el tamaño de la aplicación utilizando los procedimientos del FPA y los modelos orientados a objetos, y posteriormente, estimar el esfuerzo y la duración del proyecto con base en el tamaño estimado. Las tareas por realizar son las siguientes:

1. Modelar los casos de uso.
2. Identificar los objetos o archivos.
3. Identificar las transacciones o servicios.
4. Estimar el tamaño de la aplicación en puntos de función.
5. Estimar el esfuerzo con base en el tamaño.
6. Estimar la duración del proyecto.

4.1 Modelar los casos de uso

El modelo de casos de uso permite delimitar las fronteras de la aplicación a la que se le está estimando el tamaño. No se puede hacer un mapeo uno a uno entre los actores y los usuarios o las aplicaciones externas. Sin embargo, cualquier usuario de la aplicación representa a un actor. De la misma manera, cualquier otra aplicación que se comunique con la aplicación que se está estimando, representa un actor.

Para asegurar un mapeo consistente entre los modelos de OOSE y los procedimientos de FPA se utilizaron las siguientes reglas según Fetcke, T., Abran, A. & Nguyen, T, (1997):

1. Aceptar cada actor humano como un usuario del sistema.
2. Aceptar cada actor no humano como una aplicación externa.
3. Rechazar cualquier actor no humano que sea parte del sistema en consideración, como por ejemplo: un sistema administrador de base de datos o un dispositivo de impresión.

Para explicar la metodología se va a tomar como ejemplo el desarrollo de un "Sistema de Órdenes de Compra": En la Figura 4 se muestran los requerimientos del sistema representado en un diagrama de casos de uso. Puede verse cómo la funcionalidad está delimitada a través de los casos de uso y de los actores, que en este ejemplo son usuarios solamente, porque el sistema no interactúa con ninguna aplicación externa.

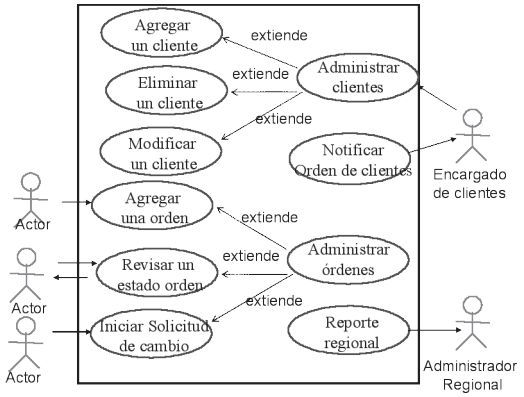


Figura 4. Modelo de casos de uso para el "Sistema órdenes de compra"

Esta tarea corresponde al paso 2 de la metodología FPA descrita en la sección 3.4 de este documento.

4.2 Identificar los objetos o archivos

Para realizar esta tarea se requiere contar con documentación del modelo de objetos del dominio o del modelo de análisis. Es probable que por ser una etapa tan temprana solo se disponga del primer modelo, es decir, solo se han identificado los objetos relevantes al dominio del problema.

Para cualquiera de los dos modelos se utilizaron las siguientes reglas recomendadas por Fetcke et al, (1997) para identificar los archivos:

Si se dispone del modelo de objetos del dominio las reglas son:

1. Seleccionar todo objeto del dominio como candidato a archivo lógico.

2. Ningún otro objeto será seleccionado.

Si se cuenta con el modelo de análisis, las reglas son:

1. Seleccionar todos los objetos de tipo entidad como candidatos a archivos lógicos.

2. Ningún otro objeto será seleccionado.

Además de aplicar las reglas anteriores, esta tarea de identificar los archivos lógicos se facilita si se utiliza la descripción de los escenarios en los casos de uso, que deberían de complementar el diagrama de casos de uso del paso anterior. Dependiendo del grado de investigación de los requerimientos, los casos de uso pueden estar descritos tal y como lo recomienda Larman, (1999), en formato de esencial o en formato expandido. Al principio se describirá en formato esencial, y conforme se obtenga más información se detallará más y se describirá en formato extendido. Pero sin importar el formato, cualquier procesamiento relacionado con la administración de datos indica la presencia de algún ILF al que hay que administrarle los datos, y cualquier procesamiento que requiera consultar información de alguna base de datos externa, indica la presencia de algún EIF.

Tomando el ejemplo del "Sistema de órdenes de compra" (modelado en la figura 4) se deberían identificar los siguientes objetos: Clientes, Órdenes de Compra y Productos.[1]

Cuadro 2. Archivos lógicos identificados en el ejemplo "Sistema de órdenes de compra"

Archivo de Órdenes	ILF
Archivo de Clientes	ILF
Archivo de Productos	ILF

Estos objetos son candidatos a ser archivos lógicos o ILF de acuerdo con la metodología del FPA. En un análisis posterior podría ser que entre Orden de Compras y Productos exista una relación de agregación, por lo que Productos no sería un ILF sino un RET, pero en una etapa tan temprana, en donde las tablas del sistema no están todavía en tercera forma normal, esta relación podría entenderse como una asociación.

Algunos datos que bajo las convenciones de FPA son archivos (ILF o EIF) puede ser que no se representen en el modelo de objetos, aunque su funcionalidad sea requerida por el usuario. Por ejemplo: mensajes de error o textos de ayuda pueden ser un requerimiento y no necesariamente son modelados como objetos.

La metodología propuesta no analiza la complejidad de los ILF o EIF desde el punto de vista de DET y RET, tal como lo propone el FPA, porque se supone que durante las primeras fases del ciclo de vida, no se conoce el detalle de los archivos como para determinar su complejidad. Esta tarea corresponde al paso 3 de la metodología FPA descrita en el punto 3.4 de este documento.

4.3 Identificar las transacciones o servicios

Para realizar esta tarea se requiere contar con documentación del modelo de casos de uso y con la descripción de los casos de uso (suficiente en formato esencial). Hay una correspondencia entre las transacciones del FPA y los casos de uso de OOSE, pero no hay una relación de uno a uno entre ambos, es decir, un caso de uso podría ser contado como una o más transacciones, dependiendo de las tareas que realice.

Para asegurar un mapeo consistente entre los modelos de OOSE y las transacciones de FPA Fetcke et al, (1997) han creado las siguientes reglas:

1. Seleccionar cada caso de uso que tenga una relación con un actor aceptado. Este caso de uso será candidato para una o más transacciones.

2. Seleccionar cada caso de uso que extienda a un caso de uso seleccionado por la regla anterior. La extensión puede incluir una interacción con un usuario o una aplicación externa. Se debe notar que las flechas en el modelo de casos de uso no indican el tipo de transacción.

3. Ningún otro caso de uso debe ser seleccionado.

Si se aplican estas reglas al ejemplo del "Sistema de órdenes de compra" se pueden identificar ocho transacciones, una por cada caso de uso, las cuales se muestran en el Cuadro 3.

Cuadro 3. Transacciones identificadas en el ejemplo "Sistema de órdenes de compra"

Función	Tipo
Agregar un cliente	EI
Modificar un cliente	EI
Eliminar un cliente	EI
Agregar una nueva orden	EI
Iniciar una solicitud de cambio	EI
Revisar el estado de una orden	EQ
Notificar orden a los clientes	EQ
Generar un informe regional	EQ

De los ocho casos de uso identificados y presentados en el Cuadro 3, los cinco primeros son EI porque actualizan el ILF de Clientes, el ILF de Órdenes de Compras y el ILF de Productos. Los tres últimos son EQ porque simplemente consultan los ILF. No se consideran EO porque hasta el momento el usuario no ha indicado que la consulta requiere algún cálculo en los datos, ni actualizan uno o más ILF que alteren el comportamiento del sistema.

Se observa cómo en el paso anterior, la metodología propuesta no analiza la complejidad de las transacciones desde el punto de vista de DET y FTR, como lo propone la metodología FPA, porque de igual manera no se cuenta con información detallada de las transacciones. Esta tarea corresponde al paso 4 de la metodología FPA descrita en el punto 3.4 de este documento.

4.4 Estimar el tamaño de la aplicación en puntos de función

En esta tarea se deben realizar los pasos 5 y 6 de la metodología FPA explicada en la sección 3.4. Con la información obtenida de los archivos (ILF y EIF) y de las transacciones (EI, EO y EQ) identificados en los pasos anteriores para el "Sistema de Órdenes de Compra", se procederá a estimar el tamaño de la aplicación. En la tabla 4 puede observarse cómo a todos los archivos y transacciones identificadas se les asigna una complejidad media. Como se indicó en los dos pasos anteriores, no se puede determinar la complejidad, porque no se dispone de información detallada por estar trabajando en una fase tan temprana del ciclo de vida, por lo que todos se consideran con una complejidad media.

Cuadro 4. Cálculo de los Puntos de Función Ajustados.

	Complejidad		
Componentes	**Baja**	**Media**	**Alta**
Archivos Lógicos Internos (ILF)	X7	3x10	x15
Archivos de Interfaz Externos (EIF)	X5	0x7	x10
Entradas Externas (EI)	X3	5 x4	x6
Salidas Externas (EO)	X4	0X5	x7
Consultas Externas (EQ)	X3	3X4	x6
Puntos de Función Sin Ajustar (PFSA)	62		
Factor de Ajuste (FA)	1,0		

Una vez obtenidos los PFSA se debe calcular el FA, pero previamente se debe haber estimado

el peso de las catorce características generales del GTI. Se asumirá que después de haberle asignado un peso a las catorce características y aplicado la fórmula (1) se obtiene un FA de 1,0 [2]. Seguidamente se aplica la fórmula (2) y se obtiene un tamaño estimado de 62 Puntos de Función Ajustados.

4.5 Estimar el esfuerzo con base en el tamaño

Una vez que se ha estimado el tamaño del software, se debe derivar el esfuerzo requerido para desarrollarlo. A pesar de que hay una larga lista de factores que influyen en la productividad del desarrollo tales como: funcionalidad solicitada, restricciones del proyecto, tecnología, métodos y ambiente de desarrollo, nivel de las habilidades y estabilidad de los requerimientos, Lawrie, (2002) propone dos enfoques de estimación importantes que se utilizan comúnmente y se describen a continuación:

1. *Estimación micro:* en este método el esfuerzo asociado con cada componente o actividad de las tareas es estimado individualmente y el resultado produce una estimación general del proyecto. Este es un enfoque "bottom-up".

2. *Estimación macro:* este método usa un enfoque "top-down". Trabaja sobre la base de promedios estadísticos. Esencialmente trata de encontrar proyectos terminados con atributos similares y extrapolar la experiencia en los nuevos proyectos. Algunos atributos que se deben considerar son: el tipo de plataforma (cliente servidor, mainframe, etc.), tipo de lenguaje (C, Java), tipo de proyecto (software de sistema, software de aplicación, etc.), tipo de sistema operativo (Windows, Unix, etc.).

La extrapolación puede ser simplemente por analogía, pero a menudo es soportada por uno o más algoritmos que producen la estimación del esfuerzo, como una función del número de variables las cuales son consideradas como "effort drivers" (factores de costos).

Respecto a la estimación macro Lawrie, (2002) indica que: "Los algoritmos que se utilizan en la estimación macro no son útiles en proyectos muy pequeños." Además, "hay un acuerdo general que indica que los proyectos muy pequeños pueden variar ampliamente en su productividad. Aunque no se ha especificado un tamaño que constituya un proyecto pequeño, muchos especialistas en métricas han acordado un límite inferior entre 50-100 PF. (Por ejemplo, Capers Jones usa 50 PF. Bankers Trust,Australia, usa 40 PF.)"

El Análisis de Puntos de Función tiene un papel importante en ambos enfoques:

Cuadro 5. Métodos de estimación para uso de los Puntos de Función

Método	Uso de los Puntos de Función
Estimación micro	El alcance del proyecto establecido como primer paso de la técnica FPA ayuda a aclarar las tareas que deben realizarse. El tamaño funcional permite calcular la tasa de productividad esperada, comparándola con datos

Fuente: (La autora)

El Cuadro 5, indica que para la estimación micro es necesario, además de conocer el tamaño funcional de la aplicación que se está estimando, previamente haber determinado la tasa de productividad promedio de la organización. Para determinarla, la organización requiere haber pasado por un proceso de recolección de métricas sobre proyectos terminados con atributos similares. Entre los atributos están: tipo de proyecto, tamaño, metas del proyecto (en cuanto a costo, calidad y tiempo), plataforma de desarrollo, lenguaje y selección de tareas (en términos de actividades y entregables asociados a esas actividades).

En cambio, en la estimación macro lo único que se requiere es conocer el tamaño funcional de la aplicación que se está estimando y aplicarlo en las fórmulas recomendadas por la industria que se indicarán más adelante.

Según Lawrie, (2002): "Típicamente los proveedores de servicios de Tecnología de Información usan la técnica de estimación micro (por tarea o por algún componente) para desarrollar la estimación del esfuerzo. Posteriormente, la técnica de estimación macro es utilizada, para validar la estimación micro. Cuando la estimación difiere de más del (10-15) % entonces se retrabaja lo estimado. No es conveniente utilizar solamente las técnicas de estimación macro, cuando se trata de contratos o licitaciones para propósitos comerciales serios."

En el curso de Ingeniería de Software de la carrera de Ciencias de la Computación e Informática en la Universidad de Costa Rica, aunque los proyectos son relativamente pequeños (aproximadamente entre 150 PF y 250 PF) se utilizan ambos enfoques por fines didácticos, aunque se le presta más atención y se le dedica más tiempo a la estimación micro. A continuación se explica la forma como se aplican ambos enfoques en dicho curso.

4.5.1 Enfoque micro (o estimación consolidada del esfuerzo)

Para estimar el esfuerzo utilizando el enfoque micro, se requiere conocer el tamaño funcional de la aplicación y la tasa de productividad de la organización, obtenida a través de bases de datos con experiencias propias de la organización y aplicarla en la siguiente fórmula:

Esfuerzo = Tamaño funcional • Tasa de productividad (3)

Donde la tasa de productividad puede expresarse como Horas/Puntos de Función.

En algunas organizaciones, para determinar la tasa de productividad asignan inicialmente un día de esfuerzo por cada Punto de Función, y a medida que se van concluyendo los proyectos se actualiza dicho valor. Si no está disponible la tasa de productividad, la organización puede ser ayudada por valores medios de la industria.

En mi caso, de acuerdo con métricas recogidas durante un año con estudiantes del curso de Ingeniería de Software se encontró que la tasa de productividad promedio de los estudiantes era de seis horas por Punto de Función, aproximadamente. En esta investigación se trabajó con ocho equipos de cinco estudiantes cada uno, desarrollando proyectos de una complejidad media. Trabajaron el ciclo de vida completo, desde la fase de conceptualización hasta la entrega. Además, los estudiantes desconocían las metodologías y herramientas que utilizaron en el desarrollo, porque eso lo aprendieron en el curso durante el año.

Continuando con el ejemplo del "Sistema de Órdenes de Compra" se tiene que la aplicación mide 62 PF. Si se asume una tasa de productividad de 6 h/PF y se aplican estos datos en la fórmula (3) se tiene:

Esfuerzo = 62 PF • 6 h/PF = 372 h persona (4)

Esto significa que una persona tardará aproximadamente 372 h desarrollando la aplicación.

4.5.2 Enfoque macro (o estimación indicativa del esfuerzo)

Una estimación indicativa o rápida usualmente se utiliza cuando hay poco tiempo e información para desarrollar sus propias métricas de productividad. Lo único que se debe hacer para estimar el esfuerzo es sustituir el tamaño obtenido en PF en las fórmulas obtenidas por la industria. A continuación se presentan dos técnicas de estimación macro recomendadas (Lawrie, 2002):

- Técnica **Capers, Jones** del SPR "Software Productivity Research" nos propone la siguiente fórmula**:**

Esfuerzo = (Tamaño en PF /150) • Tamaño en PF ^{0,4} (5)

En los algoritmos de Capers Jones el esfuerzo incluye a los desarrolladores de software, al personal de calidad, a los encargados de pruebas, a los que escriben material técnico, a los administradores de bases de datos y a los administradores del proyecto.

Siguiendo con el ejemplo del "Sistema de Órdenes de Compra", al aplicar la fórmula de Capers Jones en (5), se obtiene:

Esfuerzo = (62 PF /150) • 62 PF ^{0,4} = 2,15 meses persona = 345 h persona ^{3} (6)

Esto significa que una persona tardará aproximadamente 345 h desarrollando la aplicación.

- Técnica del **ISBSGs**. Las ecuaciones derivadas de los datos **ISBSGs**, para valores "benchmarked" como valores medios de esfuerzo para desarrollo son:

Para todos los proyectos: Esfuerzo = 11,79 • tamaño en PF ^{0,898} (7)

Para proyectos 3GL: Esfuerzo = 5,76 • tamaño en PF ^{1,062} (8)

Para proyectos 4GL: Esfuerzo = 9,32 • tamaño en PF ^{0,912} (9)

En los algoritmos del ISBCG, el esfuerzo incluye a los desarrolladores de software, administradores de proyecto y a la administración.

SE asume que el "Sistema de Órdenes de Compra" se va a programar con un Lenguaje de Cuarta generación (o 4GL) y se aplica la ecuación (9):

Para proyectos 4GL: Esfuerzo = 9,32 • 62 ^{0,912} = 401 h persona (10)

Esto significa que una persona tardará aproximadamente 401 h desarrollando la aplicación.

4.6 Estimar la duración del proyecto

Para estimar la duración de un proyecto de software se usan los siguientes factores:

1. la estimación del esfuerzo (obtenida en los pasos anteriores),
2. la plantilla de fases del ciclo de vida incluyendo el traslape entre fases y tareas,
3. la distribución del esfuerzo en las diferentes fases-tareas, y
4. la disponibilidad del personal (en cuanto a número y a tiempo).

Con esta información y una herramienta automatizada para administrar proyectos como por ejemplo: Microsoft Project, se estima la duración total del proyecto.

5. CONCLUSIONES

1. El tamaño estimado del software es muy importante en la planificación de proyectos, porque es un indicador del esfuerzo, de la duración, del costo y de los recursos requeridos (humanos, de software y de hardware) para desarrollarlo.
2. La técnica FPA requiere disponer de información muy detallada en una fase temprana, como es la conceptualización del proyecto, en donde apenas se está evaluando su viabilidad. Las reglas de conteo del FPA requieren conocer cierto nivel de detalle de los archivos y de las transacciones de la aplicación para determinar su complejidad y estimar el tamaño de la aplicación. Para identificar dicha complejidad se ha utilizado el diagrama de Entidad- Relación, el cual es típicamente una herramienta de las técnicas de diseño estructurado.
3. La dificultad de conocer información tan detallada en una etapa tan temprana como son: los atributos (DET) y los subgrupos (RET) en los archivos, y los atributos (DET) y los archivos referenciados (FTR) en las transacciones para estimar la complejidad, permite concluir que la metodología se está forzando al querer conocer tanta información en este punto y esto más bien puede generar exceso de complejidad en esta fase. Cuando se está evaluando la viabilidad del software

es suficiente con identificar los archivos ILF y EIF y las transacciones EI, EO y EQ y asignarles una complejidad media.

4. Actualmente los estudiantes utilizan métodos orientados a objetos para desarrollar, y los modelos creados con estos métodos no proveen documentación tradicional como son los diagramas de Entidad-Relación, estructuras de bases de datos o modelos de procesos jerárquicos, sino que proveen una serie de modelos que se utilizan especialmente durante las primeras fases del ciclo de vida como: el modelado de casos de uso, el modelo de objetos del dominio y el modelo de análisis.

5. A través de diferentes proyectos durante varios años, he comprobado que el estimar la complejidad de los PF y el no hacerlo, representa una diferencia relativamente pequeña, para el esfuerzo que representa estimarla en una fase temprana.

6. En una etapa tan temprana como es la conceptualización del proyecto se trabaja con requerimientos muy generales, pero no debe ser una excusa para dejar de estimar y planificar. Es importante formular el proyecto y evaluar su viabilidad para decidir si continuar o no el desarrollo. Es perfectamente válido, que después de esta evaluación se decida no implementarlo, por lo tanto, los recursos utilizados se podrían desaprovechar si se profundiza en el análisis y peor aún, en el diseño.

7. Una vez finalizada la fase de análisis y que los requerimientos se hayan concluido, es posible conocer la complejidad de los archivos y de las transacciones, y así afinar más la estimación. En este punto se conocen todos los atributos, todas las interacciones del sistema incluyendo las interfaces y reportes. Ya en este momento se puede comenzar a analizar la volatibilidad de los requerimientos y su impacto en el proyecto.

8. Tradicionalmente se ha creído que el conteo de PF no puede ocurrir hasta que el diseño esté terminado, lo cual convertiría esta metodología en algo inútil, porque se perdería uno de los principales objetivos de la planificación, el cual es determinar la duración y el costo aproximado del proyecto para evaluar su viabilidad y planificar su desarrollo, desde las primeras etapas del ciclo de vida.

9. Da acuerdo con la experiencia durantes estos años, es mejor que las organizaciones trabajen en recolectar sus propias métricas para conocer su tasa de productividad y no que tengan que usar una estimación macro, que si bien es cierto brinda información cuando no se cuenta con datos históricos, no es tan acertada.

10. El enfoque macro provee algoritmos que permiten estimar rápidamente el esfuerzo de desarrollo del proyecto con solo conocer el tamaño de la aplicación, y en general los resultados están muy cercanos al obtenido por estimación micro. De acuerdo con el ejemplo del “Sistema de Compra”, con la estimación micro se tiene que el proyecto requiere 372 h persona y por estimación macro 345 h por Capers Jones y 401 h por ISBSG. Sin embargo, no recomiendo utilizar solamente este enfoque, especialmente en proyectos pequeños, sino utilizarlos como complemento al enfoque micro.

NOTAS

1. Se asume que toda orden de compra requiere asociar los productos solicitados en la compra.

2. Para facilitar el cálculo se está considerando un FA de 1,0.

3. Asumir 160 h de trabajo por mes. Esto incluye 4 semanas, de 5 días de trabajo y 8 h laborales por día.

REFERENCIAS BIBLIOGRÁFICAS

Fetcke, T., Abran, A. & Nguyen, T. (28 Jul – 1 Aug 1997). Mapping the OO-Jacobson approach into function point analysis. *TOOLS,* 23´97. USA: Copyright 1998 IEEE.

Garmus, D. & Herron, D. (2001). *Measuring the software process. A practical guide to funcional measurements*. United States of America: Addison Wesley.

Kan, S. (2003). *Metrics and models in software quality engineering*. (2ª ed). United States of America: Adisson Wesley.

Larman, C. (1999). *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos*. México: Prentice Hall Hispanoamerica, S. A.

Lawrie, R (2002). Using functional sizing in software project estimating. *Charismatek Software Metrics*. Australia. http://www. Charismatek.com.

McConnell, S. (1997). *Desarrollo y gestión de proyectos informáticos*. España: McGraw-Hill Interamericana.

Peralta, M. (2004) Estimación del esfuerzo basada en casos de uso. [Versión electrónica]. *Reportes Técnicos en Ingeniería de Software, 6*(1), 1-16.

Pressman, R. (2006). *Ingeniería de Software: un enfoque práctico*. (6ª ed). México, D. F: McGraw-Hill Interamericana.

Quatrani, T. (2003). *Visual modeling with rational rose 2002 and UML*. United States of America: Addison Wesley.

SOBRE LA AUTORA

Gabriela Salazar Bermúdez
Universidad de Costa Rica, Escuela de Ciencias de la Computación e Informática
Apartado postal: 36-2060. San Pedro, Costa Rica.
Teléfono: 227 9544, Facsímil: 207 4020
Correo electrónico: *gsalazar@ecci.ucr.ac.cr*