Ingeniería y Ciencia
ISSN: 1794-9165
ingciencia@eafit.edu.co
Universidad EAFIT
Colombia

Salazar-B., Gabriela
Estimación de proyectos de software: un caso práctico
Ingeniería y Ciencia, vol. 5, núm. 9, junio, 2009, pp. 123-143
Universidad EAFIT
Medellín, Colombia

Ingeniería y Ciencia, ISSN 1794–9165
Volumen 5, número 9, junio de 2009, páginas 123–143

Estimación de proyectos de software: un caso práctico

A estimativa de projetos de software: um caso prático

Estimation of software projects: a practical case

Gabriela Salazar–B.[1]

Recepción: 26-ene-2009/Modificación: 29-may-2009/Aceptación: 05-jun-2009
Se aceptan comentarios y/o discusiones al artículo

Resumen

Este artículo describe una metodología para estimar y planificar proyectos de software y la experiencia en el proceso de estimación, con estudiantes del curso Ingeniería de software del programa de pregrado de la Escuela de computación en la universidad de Costa Rica. En él los estudiantes aprenden metodologías, técnicas y herramientas de ingeniería de software y desarrollan un proyecto práctico. Para estimar la duración de sus proyectos utilizan la técnica de Puntos de Función para medir el tamaño de la aplicación y, posteriormente, aplican diferentes técnicas de estimación de la duración para planificar sus proyectos. La información recopilada a través de esta investigación permite mostrar la certeza de las técnicas de estimación utilizadas, al comparar la duración estimada contra la duración real en dos hitos importantes del ciclo de vida; al inicio y al final del proyecto. Los puntos descritos en este artículo pueden interesar a líderes de proyectos, profesores e instructores que deseen formar a futuros ingenieros de software en el campo de la estimación y planificación de proyectos de software.

Palabras claves: medición, estimación, planificación, ingeniería de software.

[1] MSc, gabriela.salazar@ecci.ucr.ac.cr, profesora, Universidad de Costa Rica, San Pedro–Costa Rica.

Resumo

Este artigo descreve uma metodologia para calcular e planejar projetos de software, assim como a experiência nos processos de estimativa, com a participação de estudantes de universitário da aula de Projetos de Software na Universidade de Escola de Informática de Costa Rica. Neste curso os estudantes aprenden metodologias, técnicas e ferramentas de projetos de software e implementam um projeto prático. Para calcular a duração de seus projetos, eles usam a técnica de Pontos de Função para medir o tamanho da aplicação. Depois aplicam técnicas diferentes de estimativa da duração para planejar seus projetos. A informação colecionada desta investigação mostra o certitude das técnicas aplicadas de estimativa usadas, quando compara a duração calculada contra a duração real em dois marcos importantes do ciclo de vida (o começo e o fim do projeto). Os pontos descritos neste artigo podem interessar ás líderes, professores, e instrutores que querem formar futuros engenheiros de software na área de estimativa e planificação de projetos de software.

Palavras chaves: as medidas, estimativa, planificação, projetos de software.

Abstract

This article describes a methodology to estimate and plan software projects, as well as the experience in the estimation processes. This experience was realized with the participation of undergraduate students of the Software Engineering course at the university of Costa Rica computer science school. This course focuses on software engineering methodologies, techniques and tools and the students implement a practical project. To estimate the duration of their projects, they use the Function Points technique to measure the size of the application. Afterwards they apply different estimation techniques of the duration to plan their projects. The collected information from this investigation shows the certitude of the applied estimation techniques, when comparing the estimated duration against real duration in two important life cycle milestones (the beginning and the end of the project). The described points in this article can interest leaders, professors, and instructors that want to form future software engineers in the area of software project estimation and planning.

Key words: measurements, estimation, planning, software engineering.

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 a desarrollar

utilizando solamente el juicio de un experto, lo que produce cronogramas y presupuestos poco acertados. Con el fin de apoyar a la industria desarrolladora de software en Costa Rica, el Programa de bachillerato en computación e informática de 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. Además, es importante involucrar a los estudiantes en experiencias cercanas a la realidad de su futuro profesional.

Es importante medir el proceso de ingeniería de software y el producto que se elabora porque es la forma más objetiva de comprender y mejorar el proceso de desarrollo y el producto que se elabora. Si no se realizan mediciones, no hay forma de determinar si se está mejorando, las decisiones se basan sólo en evaluaciones subjetivas, lo que puede llevar a malas estimaciones o interpretaciones erróneas del proceso. Para establecer objetivos de mejora es necesario conocer el estado actual de desarrollo del software [1, 2].

Las métricas de software son observaciones periódicas sobre algunos atributos o aspectos del producto y proceso de software. Proveen una base para el desarrollo y validación de los modelos del desarrollo de software y pueden utilizarse para mejorar la productividad y la calidad. Por lo tanto, la medición se emplea para establecer una línea base del proceso, a partir de la cual se evalúan las mejoras. La línea base son datos recopilados en proyectos previos de desarrollo de software, que contienen medidas de proyectos y métricas derivadas de estos. Los datos de la línea base deben tener los siguientes atributos: razonablemente precisos, deben recopilarse de tantos proyectos como sea posible, las medidas deben ser consistentes y las aplicaciones que se están estimando deben ser similares [2].

En la figura 1 se ilustra el proceso con el que se obtiene una línea base de métricas. La recopilación requiere información histórica de los proyectos previos. Una vez que se han recopilado las medidas es posible calcular las métricas.

Posteriormente, las métricas se evalúan y se produce un conjunto de indicadores que guían el proyecto o proceso [2]. Estos indicadores se pueden utilizar para: evaluar la estabilidad y capacidad del proceso, interpretar los resultados de las observaciones, predecir costos y recursos para el futuro, proveer líneas base, graficar tendencias e identificar oportunidades de mejora [3].

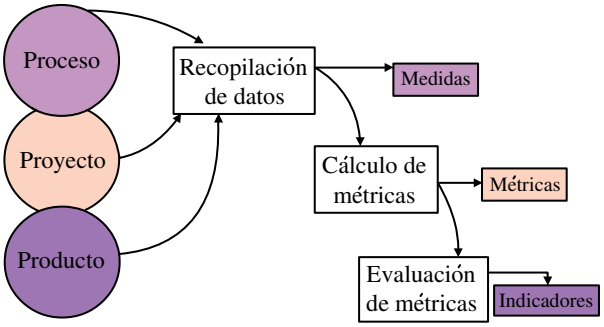

Figura 1: proceso de recopilación de métricas de software. Tomado de [2]

Una vez obtenidos los indicadores se debe ejecutar un control para verificar que el proceso se comporte consistentemente, identificar las áreas donde éste se encuentra o no bajo control, y aplicar las medidas correctivas donde sea necesario. Se debe también analizar los resultados y compararlos con promedios de proyectos similares anteriores realizados dentro de la organización, para generar conclusiones y establecer tendencias para los diferentes comportamientos [2, 3].

De acuerdo a Kan, "Las métricas de software pueden ser clasificadas en tres categorías: métricas del producto, métricas del proceso y métricas del proyecto. Las métricas del producto describen características del producto tales como tamaño, complejidad, características de diseño, rendimiento y nivel de calidad. Las métricas del proceso pueden ser utilizadas para mejorar el proceso de desarrollo y mantenimiento del software. Algunos ejemplos son efectividad de la remoción de defectos durante el desarrollo y el tiempo de respuesta en el proceso de corrección de defectos. Las métricas del proyecto describen las características y ejecución del proyecto. Algunos ejemplos son: el número de desarrolladores de software, el comportamiento del personal durante el ciclo de vida de éste, el costo, el cronograma y la productividad. Algunas métricas pertenecen a múltiples categorías. Por ejemplo, las métricas de calidad del proceso de un proyecto son ambas métricas del proceso y métricas del proyecto" [4].

En la sección 2 se explica la metodología para estimar y planificar proyectos de software. En la sección 3 se describe el curso que se utilizó como

base para realizar esta investigación y el proceso de recolección de datos. En la sección 4 se analizan los resultados de los datos obtenidos y, finalmente, en la 5, se presentan las conclusiones.

2 Metodología

En la figura 2 se muestra gráficamente la metodología que utilizan, los estudiantes en el proceso de estimación del software. La metodología está compuesta de tres tareas que se explican a continuación.

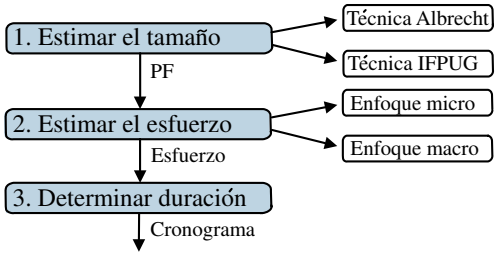

Figura 2: metodología de estimación

2.1 Estimar el tamaño

Para determinar el tamaño de la aplicación, los estudiantes utilizan el Análisis de Puntos de Función (en inglés *Function Point Analysis*, FPA). El 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 la técnica, tecnología y lenguaje de programación, por lo que introdujo los Puntos de Función (PF) como una medida del tamaño de una aplicación desde el punto de vista funcional o del usuario. Es un método que sistemáticamente se puede aplicar a cualquier tipo de software, ya sea en desarrollo o mantenimiento. Está basado en la funcionalidad del software y su

complejidad. Los PF son derivados de aspectos externos de las aplicaciones de software como: entradas, salidas, consultas, archivos lógicos e interfaces [4, 5, 6].

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* [5].

Para estimar el tamaño del software se utilizan dos técnicas: Albrecht y el IFPUG, pero en tiempos diferentes. En la figura 3 se muestran las fases en donde se recomienda aplicar dichas técnicas. La técnica de Albrecht por su sencillez (ya que no requiere conocer la complejidad de las funciones), se recomienda utilizar cuando se está conceptualizando y planificando la aplicación a nivel macro. Es conveniente estimar tempranamente toda la aplicación, para que el usuario conozca un estimado aproximado del cronograma y del presupuesto. De aquí en adelante se nombrará como "Técnica Albrecht".

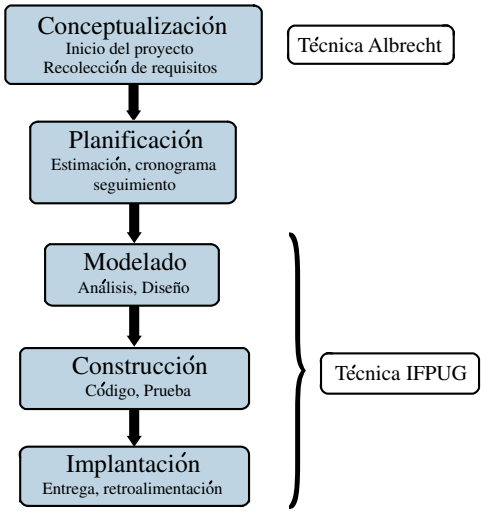

Figura 3: fases genéricas del ciclo de vida de desarrollo

Posteriormente, cuando se modele cada módulo o componente de la aplicación se podrá, entonces, refinar y actualizar la estimación del presupuesto y el cronograma definidos a nivel macro. La razón de esto es que ya se conoce el detalle de las tablas y de las transacciones porque los requerimientos están especificados. Posteriormente, esta estimación se actualiza al finalizar la fase de construcción e implantación del proyecto. Es importante que estos datos queden actualizados para futuras estimaciones.

Es importante aclarar que las fases mostradas en la figura 3 son genéricas. De acuerdo a Pressman, en [2], no importa el ciclo de vida que se utilice, se deben respetar para lograr calidad en el desarrollo del software.

A continuación se explica la técnica del IFPUG.

2.1.1 Técnicas de estimación del IFPUG. Esta metodología de medición puede resumirse en los siguientes siete pasos. Para profundizar más sobre la misma refiérase a [5].

Paso 1 (Determinar el tipo de conteo de PF)**.** Se determina el tipo de conteo de acuerdo a 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 4 se muestra gráficamente 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.

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).

El conteo físico de ILF y EIF, junto con la complejidad relativa de cada uno, determina la contribución a los PF desajustados. Esta complejidad está determinada por el Número de Elementos de Datos (en inglés *Data Element Types*, DET) y de Tipos de Registros (en inglés *Record Element Types*,

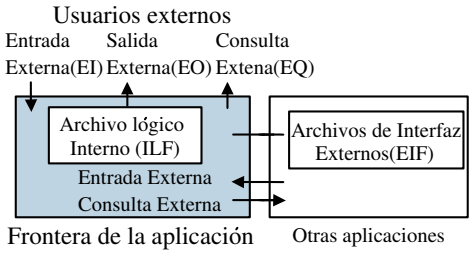

Figura 4: funcionalidad reconocida en el conteo de PF [5]

RET) asociados a cada uno. Los DET son los campos o atributos del archivo. Los RET son los grupos o subgrupos que constituyen una parte de un archivo (también conocidos como entidades débiles de una tabla).

Paso 4 (Identificar todas las funciones transaccionales y su complejidad)**.** Las transacciones transaccionales se clasifican en Entradas Externas (en inglés *External Inputs*, EI), Salidas Externas (en inglés *External Outputs*, EO) y Consultas Externas (en inglés *External Inquiries*, EQ).

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 está determinada por el número de DET y de Archivos Referenciados (en inglés *File Types Referenced*, FTR) asociados a cada transacción.

Paso 5 (Determinar los Puntos de Función Sin Ajustar (PFSA))**.** Con la información obtenida de los archivos ILF y EIF, y de las transacciones EI, EO y EQ, identificados en los pasos anteriores, se obtiene el total de PFSA. Para esto se debe emplear la tabla 1 y aplicar el peso correspondiente a la complejidad de cada transacción o archivo.

Paso 6 (Determinar el valor del Factor de Ajuste (FA)-– basado en las 14 características generales del sistema)**.** Una vez obtenidos los PFSA, se deben ajustar a través de 14 características generales, con el fin de adaptar la estimación a las condiciones de trabajo bajo las que el sistema va a ser desarrollado. A cada una de esas características se le asigna un factor de peso que indica la importancia de la característica para el sistema bajo análisis. El peso del valor asignado está entre 0 y 5: cero cuando el factor no presenta alguna influencia en la aplicación y cinco cuando el factor influye fuertemente.

Tabla 1: tabla de pesos

Componentes	**Complejidad**		
	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)

Las 14 características generales a tener en cuenta son las siguientes: comunicación de datos, procesamiento de distribuido de datos, rendimiento, configuración fuertemente utilizada, 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, y facilidad de cambio.

Luego se suman los aportes de cada característica obteniendo el Grado Total de Influencia (GTI) y se calcula el FA, utilizando la fórmula [5]

$$FA = (GTI \times 0{,}01) + 0{,}65\,. \tag{1}$$

Paso 7 (Calcular los PF ajustados)**.** Finalmente, los PF finales (ya ajustados) se obtienen como el producto de los PFSA por el factor de ajuste a través de [5]

$$PF = PFSA \times FA\,.$$

2.1.2 Técnicas de estimación de Albrecht. En esta técnica se aplican los mismos pasos del IFPUG, pero no es necesario conocer la complejidad de alguno de los componentes de la tabla 1, por lo que a todos se les aplica complejidad media. Esta es la razón por la que es muy útil aplicarla en las fases tempranas del desarrollo, en donde generalmente se conoce poca información de la aplicación a desarrollar.

A continuación se explican las técnicas de estimación del esfuerzo que se utilizaron en esta investigación. Estas técnicas requieren como parámetro de entrada los PF.

2.2 Estimar el esfuerzo

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, en [7], propone dos enfoques de estimación importantes que se utilizan comúnmente y se describen a continuación:

1. Estimación micro: este método usa una lista de tareas y estructura de trabajo para identificar los elementos, los cuales son estimados independientemente usando métodos y técnicas apropiadas. Este es un enfoque *bottom–up*.

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

El análisis de PF tiene un papel importante en ambos enfoques:

Método	**Uso de los PF**
Estimación micro	El tamaño funcional permite calcular la tasa de productividad esperada, comparándola con datos históricos de la organización.
Estimación macro	El tamaño funcional es una entrada clave en los algoritmos o fórmulas de estimación.

Según Lawrie, en [7], "típicamente los proveedores de servicios de tecnología de información usan la técnica de estimación micro (por tarea o por

algún componente) para estimar el 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, aunque los proyectos son relativamente pequeños (aproximadamente entre 230 PF y 350 PF), se utilizan ambos enfoques por fines didácticos, pero se le presta más atención y se le dedica más tiempo a la estimación micro. A continuación se explican ambos enfoques y la forma como se aplican en el curso.

2.2.1 Estimación micro. Para estimar el esfuerzo en este enfoque se requiere conocer el tamaño funcional de la aplicación (obtenido por la Técnica FPA) y la tasa de productividad de la organización. La productividad significa el número de horas que ocuparon para implementar un PF. Para determinar la tasa de productividad 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). Se aplica la fórmula

$$\text{Esfuerzo} = \text{Tamaño funcional} \times \text{Tasa de productividad}\,, \tag{2}$$

donde la tasa de productividad puede expresarse como horas/PF.

Para obtener el parámetro "tasa de productividad" de (2), algunas organizaciones asignan inicialmente un día de esfuerzo por cada PF y a medida que se van concluyendo proyectos, van creando un repositorio que les permita actualizar dicho parámetro para ajustarlo. Otra manera de obtener la tasa de productividad, si no está disponible en la organización, es utilizar valores medios de la industria.

En el curso se inició con una tasa de productividad de ocho horas por PF y año tras año se ha ido ajustando, gracias al proceso de recolección de métricas que se ha venido realizando durante los últimos seis años. Actualmente la tasa de productividad está aproximadamente en 4,27 horas/PF.

2.2.2 Estimación macro. 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 por Lawrie en [7]:

1. Capers Jones del *Software Productivity Research* (SPR) propone

$$\text{Esfuerzo} = (\text{Tamaño en PF}/150) \times \text{Tamaño en PF}^{0,4}\,. \qquad (3)$$

   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.

2. Las ecuaciones derivadas de los datos **ISBSGs**, para valores *benchmarked* como valores medios de esfuerzo para desarrollo son:

$$\begin{array}{ll} \text{Para todos los proyectos} & \text{Esfuerzo} = 11{,}79 \times \text{tamaño en PF}^{0,898}\,, \\ \text{Para proyectos 3GL} & \text{Esfuerzo} = 5{,}76 \times \text{tamaño en PF}^{1,062}\,, \\ \text{Para proyectos 4GL} & \text{Esfuerzo} = 9,32 \times \text{tamaño en PF}^{0,912}\,. \end{array} \qquad (4)$$

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

2.3 Determinar la duración

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

1. La estimación del esfuerzo (explicada en la sección 2.2),
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).

El esfuerzo estimado se debe distribuir entre las actividades del ciclo de vida, tomando como base el paradigma seleccionado (proceso de desarrollo unificado, metodología ágil, espiral, etcétera). De acuerdo al paradigma hay que considerar la secuencia y traslape entre tareas. Para distribuir el esfuerzo entre las actividades se pueden utilizar los porcentajes de distribución que recomienda Pressman en [5] y se muestran en la figura 5.

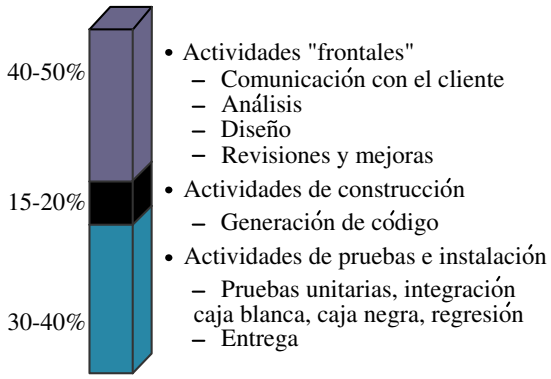

Figura 5: distribución recomendada por Pressman 40–20–40

Con esta información y una herramienta automatizada para administrar proyectos, se define la duración total del proyecto y la fecha final aproximada.

Otro método para obtener la duración del proyecto es el denominado algorítmico COCOMO II, el cual consiste en la aplicación de ecuaciones matemáticas sobre los PFSA, o sobre las líneas de código estimadas para un proyecto. Estas ecuaciones se encuentran ponderadas por ciertos *cost drivers* que influyen en el esfuerzo requerido para el desarrollo del software. La técnica de COCOMO II se sale del alcance de este artículo, pero se puede encontrar información en [4].

En la sección 3 se describe el curso y el ambiente de trabajo en el que los estudiantes realizan sus proyectos.

3 Descripción del curso

En el Programa de bachillerato de computación e informática en la Universidad de Costa Rica, la ingeniería de software se imparte a través de los cursos Ingeniería de software I e Ingeniería de software II, con sus respectivos laboratorios (ver figura 6). Los cursos son semestrales, uno es requisito del otro y cada uno tiene una duración aproximada de 16 semanas lectivas. Específicamente, Ingeniería de software I y II son cursos teóricos en donde se enseña metodologías, técnicas y herramientas de ingeniería de software. Se imparten en cuatro horas lectivas semanales y valen cuatro créditos cada uno. Por otro lado, en los cursos de laboratorio los estudiantes aplican los conocimientos teóricos; se ofrecen en dos horas lectivas semanales y valen un crédito cada uno.

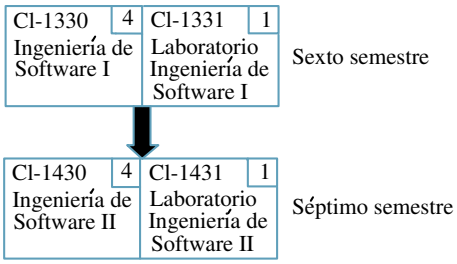

Figura 6: secuencia de cursos de Ingeniería de software en el programa de estudios

Durante los dos semestres que dura el curso los estudiantes desarrollan la misma aplicación. Al inicio del año se les dio una definición básica de los requerimientos del proyecto, la cual debían completar y mejorar a través de entrevistas y otras técnicas como prototipado. Esto hacía que cada proyecto, aunque partiera de una misma definición, variara tanto en el diseño de las interfaces de entrada y salida como a nivel de estructuras de datos. El objetivo es fomentar la creatividad en ellos pero con ciertas restricciones, lo cual produjo variaciones en el tamaño de las aplicaciones según el equipo que la desarrolló [2].

Las características de los proyectos eran muy similares: el mismo problema a resolver, como plataforma operativa utilizaron Microsoft.NET, el lenguaje de programación fue ASP.NET y C#, el sistema administrador de base de

datos que se utilizó fue SQL server, la herramienta de análisis y diseño para hacer el modelado en UML fue Rational Rose, y como herramienta para administrar el proyecto se usó Microsoft Project. Se implementó en la arquitectura N capas, utilizando el proceso de desarrollo unificado (iterativo e incremental).

Cada fase del ciclo de vida durante el proceso de desarrollo era guiada por un estándar adaptado a las características del curso, pero siguiendo las prácticas internacionales recomendadas por el IEEE [8] y por el PMBOK Guide [9]. Lo que se pretende con el uso de estándares en el curso es que todos trabajen en forma estandarizada, aplicando prácticas y metodologías de calidad internacionales para obtener un producto final de calidad.

La recolección de datos se realizó durante el año 2006. Se conformaron seis equipos de trabajo compuestos por cuatro estudiantes cada uno incluyendo el líder. Ningún estudiante tenía experiencia en el desarrollo de aplicaciones, ni utilizando las metodologías y herramientas que se enseñan en el curso.

El proceso de recolección de datos se efectuó de la siguiente manera:

1. Cada miembro del equipo de trabajo debía reportar a su líder el trabajo individual a través de una minuta, indicando la tarea realizada y la duración correspondiente.

2. De igual manera, las reuniones de grupo tenían que reportarse en una minuta, en donde se especificaba: la tarea, los miembros presentes y la duración de la reunión.

3. Al finalizar cada fase, el líder, como representante del equipo, debía presentar un informe especificando un resumen por fase y por miembro, y adjuntando las minutas correspondientes a la fase para verificar sus resultados.

A continuación se muestran los resultados obtenidos durante esta experiencia.

4 Presentación y análisis de resultados

En las tablas 2 y 3 se muestran los resultados obtenidos a través de esta investigación y en las figuras 7 y 8 los datos graficados.

Tabla 2: datos estimados durante la fase de conceptualización versus duración real

Proyecto	Tamaño PF–Técnica Albrecht	Enfoque macro		Enfoque micro	
		Técnica Capers Jones	Técnica ISBSG	Técnica Albrecht	Duración real
A	276	1188,77	1157,11	1181,25	1086
B	266	1126,68	1380,75	1136,84	636
C	241	985,28	1266,94	1033,00	869
D	239	969,80	1254,14	1021,38	1003
E	262	1102,11	1361,36	1119,08	807
F	260	1089,88	1351,65	1110,20	853
Promedio	**257,33**	**1077,09**	**1295,33**	**1100,29**	**875,58**

La columna "Tamaño PF–Técnica Albrecht" mide la aplicación durante la fase de conceptualización. Las columnas "Técnica Capers Jones", "Técnica ISBSG" y "Enfoque micro Técnica Albrecht" muestran la duración estimada en horas, obtenidas durante la fase de conceptualización utilizando (3), (4) y (1), correspondientemente. En "Enfoque micro Técnica Albrecht" se utilizó una productividad promedio de 4,27 horas/PF. La columna "Duración real" corresponde al total de horas que los estudiantes duraron implementando la aplicación. Incluye el acumulado de horas desde la fase de conceptualización hasta la fase de entrega.

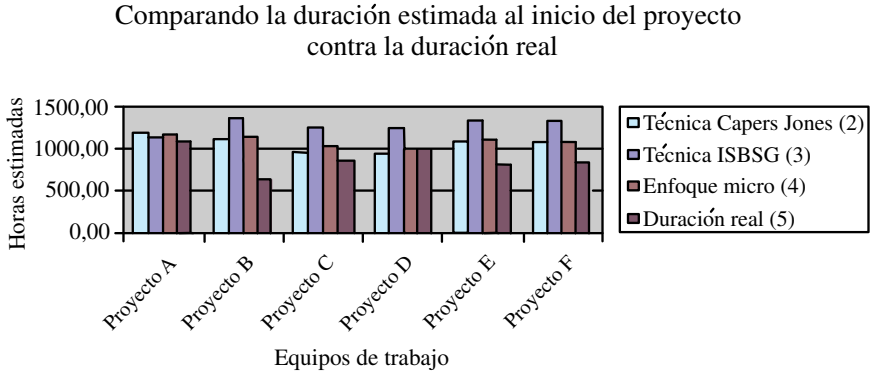

Figura 7: comparación de las estimaciones durante la fase de conceptualización

Se puede observar que, en general, las tres técnicas de estimación utilizadas muestran valores muy cercanos entre sí con la duración real, pero la que más se acerca a la duración real es la técnica de Albrecht, que es del enfoque micro. Hay dos aspectos importantes de resaltar:

1. La aproximación de las estimaciones con la duración real demuestra la efectividad de las técnicas, especialmente porque se aplican en la fase de conceptualización del proyecto, en un momento en donde se cuenta con poca información. Estas estimaciones permiten realizar una planificación del proyecto a nivel macro, y muy cercana a la realidad.

2. El que la técnica de Albrecht sea la que más se acerca a la duración real está justificado, porque esta técnica utiliza la productividad real de los estudiantes en un determinado ambiente de trabajo, en cambio, las del enfoque macro usan productividades de la industria. Sin embargo, las técnicas del enfoque macro sólo se requieren como parámetro de entrada del tamaño de la aplicación, en cambio, el enfoque micro requiere además del tamaño, la productividad, que no es un dato fácil de obtener porque se necesita un repositorio con datos históricos de proyectos concluidos. Esto significa que si no se conoce la productividad, una forma rápida de estimar es utilizando las técnicas del enfoque macro.

Tabla 3: datos estimados al inicio y final del proyecto versus duración real

Proyecto	Tam. PF– Técnica Albrecht	Enf. micro Técnica Albrecht	Tam. PF– Técnica IFPUG	Enf. micro Técnica IFPUG	Duración real	Productividad real
A	276	1181,25	227	972,54	1086	4,78
B	266	1136,84	191	817,11	636	3,33
C	241	1033,00	187	799,34	869	4,65
D	239	1021,38	198	848,19	1003	5,07
E	262	1119,08	200	852,63	807	4,04
F	260	1110,20	227	972,54	853	3,76
Promedio	**257**	**1100,29**	**205**	**877,06**	**875,58**	**4,27**

Las columnas “Tamaño PF–Técnica Albrecht” y “Tamaño PF-–Técnica IFPUG” presentan el tamaño de la aplicación medido en PF. La diferencia

entre ambas técnicas es que la primera fue obtenida durante la fase de conceptualización y la segunda estimada en la fase de análisis, pero actualizada al finalizar el proyecto. Las columnas "Enfoque micro Técnica Albrecht" y "Enfoque micro Técnica IFPUG" muestran la duración estimada en horas, la primera obtenida durante la conceptualización con la técnica Albrecht, y la segunda obtenida al finalizar el proyecto con la técnica del IFPUG. En ambas se aplicó (1) con una productividad promedio de 4,27 horas/PF. La columna "Duración real" muestra el total de horas que los estudiantes duraron implementando la aplicación. Incluye el acumulado desde la fase de conceptualización hasta la de entrega. La columna "Productividad real" muestra la productividad de cada equipo de desarrollo. Se obtuvo a través de la división de la duración real entre el tamaño de la aplicación en PF.

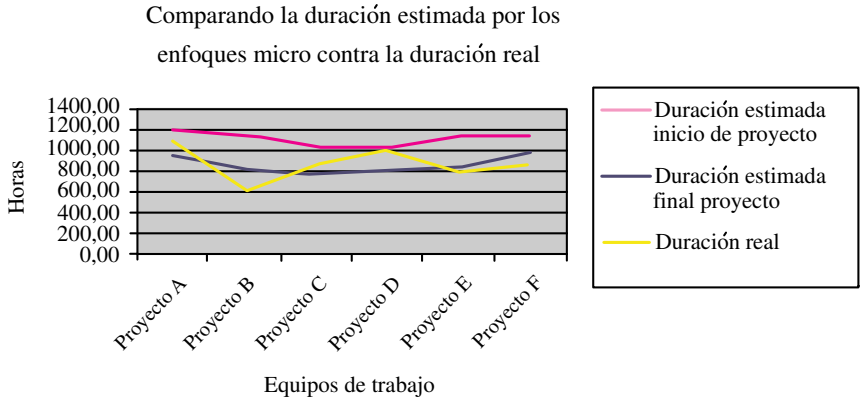

Figura 8: comparación de las estimaciones contra la duración real

La duración real es más cercana a la estimada con la técnica del IFPUG, que con la técnica de Albrecht. Esto es justificado porque la de Albrecht maneja complejidades media de las transacciones y de las tablas, en cambio, con la del IFPUG se determina la complejidad de estos elementos. Por otro lado, la técnica de Albrecht se aplica en la fase de conceptualización donde se dispone de poca información de la aplicación, y la del IFPUG durante el desarrollo y se actualiza una vez implementada.

Otro aspecto importante de observar es que el tamaño de la aplicación tiende a disminuir y a acercarse más a la realidad conforme avanza el desarrollo

del proyecto, y es que, conforme se progresa, se cuenta con más información y se conoce más la aplicación. Esto es positivo porque es mejor que en la fase de conceptualización, el presupuesto y el cronograma estimado estén holgados y no muy ajustados.

5 Conclusiones

De acuerdo a los resultados arrojados en esta investigación, se puede concluir que:

- Para estimar la duración de un proyecto, una organización que no cuente con métricas de productividad, puede iniciar su proceso de estimación utilizando la técnica de Albrecht para determinar el tamaño de la aplicación, y con base en este parámetro aplicar las técnicas de enfoque macro para estimar la duración del proyecto.

- La técnica de Albrecht tiene la ventaja, respecto a la del IFPUG, que no requiere conocer la aplicación en forma detallada porque determina una complejidad media para las tablas y las transacciones. Por otro lado, el enfoque macro tiene la ventaja sobre el enfoque micro de que no requiere conocer la productividad de la organización. Sin embargo, como se puede observar en los resultados descritos en la sección anterior, cuando se utiliza la técnica del IFPUG y el enfoque micro, la duración estimada se acerca más a la duración real, por lo que es más certera.

- Por lo tanto, se recomienda que toda organización cuente con un proceso de recolección de métricas que le permita conocer su productividad de desarrollo. No obstante, mientras se recogen métricas se puede utilizar el enfoque micro utilizando una productividad entre seis y ocho horas por PF, dependiendo de la complejidad de la aplicación que se está estimando. Y una vez recogidos datos históricos, de tamaños y duraciones de aplicaciones concluidas, se actualiza el parámetro de la productividad.

- Aunque la técnica de IFPUG es más tediosa que la de Albrecht y requiere conocer más información de la aplicación, se recomienda su uso

porque los resultados demuestran que el tamaño estimado con esta técnica se acerca más al tamaño real y éste es un parámetro indispensable para estimar la duración, ya sea con el enfoque micro o con el macro.

- Algunos beneficios de la estimación son: plazos de entrega y presupuesto más realistas, mayor satisfacción de los usuarios, cronogramas más acertados que permiten controlar mejor el proyecto, los procesos de estimación son una exigencia de los estándares de calidad internacionales y, por último, al mejorar la industria de desarrollo de software, el país puede competir con mercados internacionales.

- Con la experiencia de estos años en el proceso de recolección de métricas se pueden resumir dos grandes beneficios: primero, se involucra a los estudiantes en experiencias cercanas a la realidad de su futuro profesional y segundo, se les prepara en las nuevas tendencias de calidad, en donde las métricas juegan un papel muy importante.

- En un futuro es importante trabajar con el gobierno y la empresa privada para apoyarlos en el área de estimación y planificación de proyectos de software. También es conveniente experimentar con otras técnicas de estimación para validar y comparar los resultados de esta investigación.

Reconocimientos

Un agradecimiento especial a los estudiantes del curso de Ingeniería de software del año 2006, que participaron en el proceso de recolección de métricas.

Referencias

[1] E. Mills. *Software Metrics SEI Curriculum Module SEI–CM–12–1.1*, ftp://ftp.sei.cmu.edu/pub/education/cm12.pdf, diciembre 1988. Referenciado en 125

[2] Roger S. Pressman. *Ingeniería de software: un enfoque práctico*, 6$^{\text{ta}}$ edición, ISBN 970–10–5473–3. McGraw–Hill, 2005. Referenciado en 125, 126, 129, 136

[3] William A. Florac and Anita D. Carleton. *Measuring the software process: statistical process control for software process improvement*, ISBN 0–201–60444–2. Addison–Wesley Professional, 1999. Referenciado en 125, 126

[4] Stephen H. Kan. *Metrics and models in software quality engineering*, second edition, ISBN 0–201–72915–6. Adisson wesley, 95–456 (2002). Referenciado en 126, 128, 135

[5] Garmus and David Herron. *Measuring the software process. A practical guide to funcional measurements*, ISBN 0–201–69944–3. Addison wesley, 2001. Referenciado en 128, 129, 130, 131, 135

[6] Richard D. Stutzke. *Estimating software—intensive systems*, ISBN 0–201–70312–2. Addison–Wesley Professional, 2005. Referenciado en 128

[7] R. Lawrie. *Using functional sizing in software projects estimating.* http://www.charismatek.com, Charismatek software metrics, Australia, 2002. Referenciado en 132, 134

[8] IEEE. *IEEE Software Engineering Standards Collection: 2003*, ISBN 978–0738137575. Inst of Elect & Electronic, 2003. Referenciado en 137

[9] Project Management Institute. *Guide to the project management body of knowledge (PMBOK guide)*, ISBN 1–880410–23–0. Third edition, 2005. Referenciado en 137