Una Experiencia con Estudiantes para la Estimación del Esfuerzo de cada Iteración en Proyectos de Software

José Antonio Pow-Sang Portillo
Pontificia Universidad Católica del Perú, Dpto. de Ingeniería,
Lima, Perú
japowsang@pucp.edu.pe

Abstract

Effort and cost estimation is still one of the hardest tasks in software project management. At the moment, there are many techniques to do this job, like Function Points, Use Case Points and COCOMO, but there is not much information about how to use those techniques in non-waterfall software lifecycles such as iterative or spiral lifecycles projects.
This paper shows the results obtained when applying a technique to estimate the effort of each construction iteration in software development projects that use iterative-incremental lifecycles. The results were obtained from software projects of a fourth-year course in Informatics. The students calculated the effort of the second and third iteration of construction on the basis of the results obtained in previous iterations. The technique proposes the use of Function Points and COCOMO II and it has been proved in a previous semester.

Keywords: Effort Estimation, iterative-incremental lifecycle, software engineering experimentation, use cases.

Resumen

La estimación del esfuerzo y costo es una de las tareas más difíciles en la gestión de proyectos de software. En la actualidad, existen técnicas para realizar esta tarea. Algunas de ellas son: Puntos de Función, Puntos de Casos de Uso y COCOMO II. Lamentablemente, no hay mucha información de cómo utilizar las técnicas en mención para ciclos de vida diferentes al de cascada, como los ciclos de vida iterativo-incrementales y en espiral.
Este artículo muestra los resultados obtenidos en la aplicación de una técnica para estimar el esfuerzo de proyectos que usan los ciclos de vida iterativo-incrementales. Estos resultados fueron obtenidos en proyectos realizados por alumnos de cuarto año de la carrera de Ing. Informática. Para esta actividad, los estudiantes calcularon el esfuerzo de la segunda y tercera iteración de la fase de construcción tomando como base los resultados que obtuvieron en iteraciones previas. La técnica propone el uso de Puntos de Función y COCOMO II y ha sido probada en un semestre anterior al de esta experimentación.

Palabras claves: Estimación del esfuerzo, ciclo de vida iterativo-incremental, experimentación en Ing. de Software, casos de uso.

1. INTRODUCCIÓN

La creciente complejidad de los desarrollos software que provocó la denominada "crisis del software" se ha tratado de abordar mediante el planteamiento de nuevos métodos, metodologías, técnicas y paradigmas para minimizar su impacto. El alcance de dichas propuestas no se limita exclusivamente a actividades relacionadas con el desarrollo en sí de los sistemas, sino que abarca también las actividades de gestión de los mismos. Una de estas actividades es la de la estimación de los proyectos software.
A pesar de que la estimación de proyectos continúa siendo una tarea muy compleja, en muchas ocasiones dejada al albur de la pericia del experto estimador, en las últimas décadas se han desarrollado algunas técnicas para la estimación del esfuerzo de proyectos software completos, tales como Puntos de Función [23], Puntos de Caso de Uso [10] y COCOMO II [4]. Aún así, estas técnicas –si bien se postulan como independientes de la tecnología final de desarrollo– fueron concebidas para su aplicación en sistemas basados en el paradigma estructurado con un ciclo de vida clásico o en cascada de Royce [18], y aún es difícil emplearlas en desarrollos orientados a objetos y ciclos de vida iterativo-incrementales, tan en boga en los últimos años. Incluso, parece interesante que éstas técnicas de estimación exploten

para sus propósitos la información proporcionada por prácticas muy extendidas últimamente, como, por ejemplo, la de los casos de uso.
Conociendo las ventajas de realizar experimentos con alumnos [5] y teniendo en cuenta la problemática planteada, el presente artículo muestra una experiencia en la aplicación de una técnica basada en Puntos de Función (PF) que propone un cálculo de estimación del esfuerzo de cada una de las iteraciones. Esta experiencia fue realizada para dos equipos de trabajo conformados por alumnos de 4to año de la carrera de Ing. Informática, en el semestre 2004-2 (agosto-diciembre 2004), dentro de un curso del área de Ing. de Software.
Este documento se ha estructurado de la siguiente manera: la sección 2 muestra un breve resumen de las técnicas de Puntos de Función, COCOMO II y su relación con los casos de uso; la sección 3, la técnica para estimar y planificar la construcción de software para ciclos de vida iterativos-incrementales; la sección 4, la información correspondiente a la experimentación; la sección 5, los resultados obtenidos; la sección 6, una discusión de los resultados obtenidos con ambas técnicas; y, finalmente, se presentan las conclusiones y trabajo futuro de esta experiencia..

2. LAS TÉCNICAS DE ESTIMACIÓN Y LOS CASOS DE USO

Esta sección muestra un breve resumen de las técnicas de Puntos de Función y COCOMO II, y su relación con los casos de uso. Además, se da una visión general del trabajo que se ha encontrado en relación al tema.

2.1 La Técnica de Puntos de Función.

Los Puntos de Función [20] fueron introducidos por Albrecht [1] y su propósito es medir el software cualificando la funcionalidad que proporciona externamente, basándose en el diseño lógico del sistema. Los objetivos de los Puntos de Función son:

- Medir lo que el usuario pide y lo que el usuario recibe.
- Medir independientemente de la tecnología utilizada en la implantación del sistema.
- Proporcionar una métrica de tamaño que dé soporte al análisis de la calidad y la productividad.
- Proporcionar un medio para la estimación del software.
- Proporcionar un factor de normalización para la comparación de distintos software.

El análisis de los Puntos de Función se desarrolla considerando cinco parámetros básicos externos del Sistema: External Input (EI), External Output (EO), External Query (EQ), Internal Logic File (ILF) y External Interface File (EIF).
Con estos parámetros, se determinan los puntos de función sin ajustar (PFsA). A este valor, se le aplica un Factor de Ajuste obtenido en base a unas valoraciones subjetivas sobre la aplicación y su entorno; es decir, las características generales del sistema.

2.2 COCOMO II y los Puntos de Función

COCOMO II, propuesto y desarrollado por Barry Boehm [4], es uno de los modelos de estimación de costos mejor documentados y utilizados. El modelo permite determinar el esfuerzo y tiempo que se requiere en un proyecto de software a partir de una medida del tamaño del mismo expresada en el número de líneas de código que se estimen generar para la creación del producto software.
Debido a la complejidad de los proyectos de software, el modelo original fue modificado, denominándose al modelo actual COCOMO II. El nuevo modelo permite estimar el esfuerzo y tiempo de un proyecto de software en dos etapas diferentes: diseño temprano y post-arquitectura. Para ambas etapas, el modelo propone usar un conjunto de drivers de coste, los cuales indican el contexto en el cual se está desenvolviendo el proyecto. Estos drivers son utilizados para determinar el factor de ajuste "EAF", el cual se utiliza para realizar el cálculo del esfuerzo necesario para completar el proyecto. La tabla 1 muestra la relación de los drivers correspondientes al modelo de post-arquitectura.

Tabla 1: Drivers de coste para el modelo de Post-Arquitectura de COCOMO II

Driver de Coste	Descripción
RELY	Fiabilidad requerida del software
DATA	Tamaño de la base de datos
CPLX	Complejidad del producto
RUSE	Reusabilidad requerida
DOCU	Documentación de acuerdo a las necesidades del ciclo de vida
TIME	Restricción de tiempo de restricción
STOR	Restricción de almacenamiento principal
PVOL	Volatilidad de la plataforma
ACAP	Capacidad de analistas
PCAP	Capacidad de programadores

PCON	Continuidad del personal
AEXP	Experiencia en aplicaciones
PEXP	Experiencia de plataforma
LTEX	Experiencia de lenguajes y herramientas
TOOL	Uso de herramientas de software
SITE	Desarrollo en múltiples lugares

Otro cambio realizado en el nuevo modelo consiste en poder determinar el esfuerzo y tiempo de un proyecto de software a partir de los puntos de función sin ajustar (PFsA), lo cual supone una gran ventaja, dado que en la mayoría de los casos es difícil determinar el número de líneas de código de que constará un nuevo desarrollo, en especial cuando se tiene poca o ninguna experiencia previa en proyectos de software. Esto hace que ambos modelos –Puntos de Función y COCOMO II– sean perfectamente compatibles y complementarios.

2.3. Los Casos de Uso y los Puntos de Función.

Los casos de uso fueron introducidos en 1987 como una herramienta de la técnica Objetory [19]. Su utilización en los procesos de Ingeniería de Software fue propuesta por Ivar Jacobson y publicada en su libro "Object Oriented Software Engineering" [9]. Actualmente, su empleo se ha extendido aún más, debido a su inclusión en UML [12], por lo que su uso en el desarrollo de software orientado a objetos se ha vuelto altamente recomendable, si no obligatorio.

David Longstreet, en uno de sus artículos [11], señala que el análisis de puntos de función se puede aplicar de manera sencilla con los casos de uso mejorando la calidad de los documentos de requerimientos y, a la vez, mejorando la estimación del proyecto de software. El aplicar la técnica de Puntos de Función permite verificar y validar el contenido de un documento de Especificación de Requisitos de Software.

Lo interesante de la aplicación de ambas técnicas es que se puede actualizar el conteo de los puntos de función cada vez que los casos de uso cambien y, de esta manera, determinar el impacto que un caso de uso específico puede producir en la estimación del desarrollo de todo el proyecto.

Thomas Fetcke [7] propone una forma para utilizar la técnica de puntos de función con la metodología OOSE de Jacobson [9]. La relación que muestra se basa en los casos de uso y en el diagrama de clases de análisis propuestos por dicha metodología. En su propuesta no especifica el tipo de ciclo de vida que se debe seguir con su técnica, por lo que se podría inferir que se debería usar el modelo ciclo de vida en cascada.

3. TÉCNICA PARA PLANIFICAR Y ESTIMAR LAS ITERACIONES DE UN PROYECTO DE SOFTWARE CON PUNTOS DE FUNCIÓN

En [13] se propuso una técnica para estimar y planificar las iteraciones en base a puntos de función y COCOMO II y es la que se utilizó para esta experimentación.

La técnica en mención propone dos pasos principales: en primer lugar, determinar los casos de uso a realizar por iteración y, posteriormente, estimar el esfuerzo para cada iteración. A continuación se detallan las actividades que se siguen en cada uno de los pasos mencionados.

3.1 Determinar los Casos de Uso a Realizar por Iteración (paso 1).

El objetivo de esta tarea es determinar los casos de uso que se deberán implementar en cada iteración. Para ello, se deberá haber realizado previamente la especificación de todos los casos de uso del software a construir y que deberán estar en el documento de especificación de requisitos de software (para la especificación de casos de uso, se puede tomaron las recomendaciones incluidas en [2] [3][14][22] y para el documento de especificación de requisitos de software, [8][17]).

Para esta actividad se propone la utilización de un nuevo diagrama al que se ha denominado "diagrama de precedencias", en el cual se muestran gráficamente las precondiciones de cada uno de los casos de uso que se incluyen en sus respectivas especificaciones. La idea de dicho diagrama fue tomada de Doug Rosengberg [20] quien propone la realización de un diagrama similar al propuesto, especificando las relaciones "precede" e "invoca" (invoke en inglés) para determinar los requerimientos de usuario. La Figura 1, muestra un ejemplo de un diagrama de precedencias.

El diagrama se utilizará para saber qué caso de uso necesita de alguna funcionalidad o información que es administrada o implementada por otro caso de uso. Esto permitirá conocer qué caso de uso debe programarse antes que otro, de manera que lo necesario para que se pueda implementar un caso de uso ya haya sido desarrollado en una iteración previa.

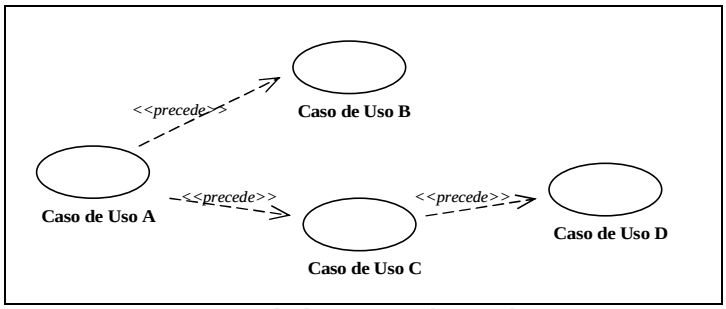

Figura 1. Ejemplo de Diagrama de Precedencias

Siguiendo la idea del párrafo anterior, los casos de uso que se encuentran a la izquierda del diagrama, se implementarán antes de los que se encuentran a la derecha; es decir, en el ejemplo propuesto en la Figura 1, "Caso de uso A" se deberá implementar antes que "Caso de uso C".
Es importante resaltar que en este diagrama no se consideran los casos de uso incluidos y extendidos, ya que éstos se pueden considerar como parte de aquellos que les hacen referencia.

3.2 Estimación del Esfuerzo para Cada Iteración (paso 2).
La siguiente actividad, después de determinar qué caso de uso se realizará en cada iteración, consiste en determinar el esfuerzo que tomará cada iteración. Para ello, se propone la utilización de las técnicas de puntos de función y COCOMO II.
La primera tarea consiste en determinar los puntos de función sin ajustar para cada iteración. El criterio seguido, para adaptar la técnica de puntos de función para ciclos de vida iterativos-incrementales, consiste en considerar que el resultado del cálculo de los puntos de función sin ajustar para todo el proyecto y sin considerar iteraciones (PFsA_Total), deberá ser igual a la suma de los puntos de función sin ajustar calculados para cada iteración por separado (*PFsA(i)* = puntos de función sin ajustar de la iteración "i").

$$PFsA\_Total = \sum_{i=1}^{n} PFsA(i)$$

Uno de los aportes más importantes de esta propuesta consiste en determinar los puntos de función sin ajustar correspondientes a los ficheros lógicos internos (ILF) y ficheros de interfaz externa (EIF) para cada caso de uso. Para ello, esta técnica propone utilizar la siguiente fórmula (*Fich_PFsA* = punto de función sin ajustar para un caso de uso "j" debido a los ficheros ILF/EIF, *TCU(i)* = número total de casos de uso que utiliza un ILF/ EIF "i", *Peso(i)*= Peso debido a la complejidad del ILF/EIF "i", *i*= ILF/ EIF utilizado en el caso de uso "j" y *j*= caso de uso en cuestión)

$$Fich\_PFsA(j) = \sum_{i=1}^{n} \frac{1}{TCU(i)} xPeso(i)$$

Con los resultados obtenidos con la fórmula anterior y sabiendo qué caso de uso se desarrollará en cada iteración, se determinan los PFsA debido a los ficheros por iteración. A esto se le añadirán los PFsA correspondientes a las transacciones. Para ello, se utiliza la fórmula que se muestra a continuación (*i*= iteración específica, *TPFsA(i)*= total de puntos de función sin ajustar para una iteración "*i*", *Fich_PFsA(j)* = punto de función sin ajustar para un caso de uso "*j*" debido a ILF/EIFs, *Trans_PFsA(j)* = punto de función sin ajustar para un caso de uso "*j*" debido a las transacciones (EI,EO,EQ) y *j*= caso de uso a implementar en una iteración "*i*").

$$TPFsA(i) = \sum_{j=1}^{n} \left[Fich\_PFsA(j) \right] + \sum_{j=1}^{n} \left[Trans\_PFsA(j) \right]$$

Luego, utilizando COCOMO II y con los puntos de función sin ajustar correspondientes a cada iteración, se obtiene el esfuerzo en meses-hombre de cada una de las iteraciones y, con este valor, se puede determinar el tiempo y personas necesarias para acabar la iteración y por ende el tiempo de todo el proyecto.
Es importante resaltar que el contexto del proyecto puede cambiar al pasar de una iteración a otra (conocimiento de la plataforma de desarrollo, integración del equipo de desarrollo, etc), por lo que podría ser necesario reestimar de nuevo el esfuerzo requerido para las iteraciones siguientes, revisando ficheros (ILF/EIF) y transacciones (EI, EO y EQ) en caso de cambio de requisitos, y recalculando drivers de coste propuestos por COCOMO II en caso de cambios contextuales u organizacionales.

4. DISEÑO DE LA EXPERIMENTACIÓN

Los estudiantes que participaron en esta experimentación fueron alumnos de cuarto año de la carrera de Ing. Informática de la Pontificia Universidad Católica del Perú. Esta experimentación se desarrolló en una asignatura obligatoria del área de Ingeniería de Software y como parte del curso los alumnos tenían que realizar un proyecto de desarrollo de software. La duración fue de 14 semanas correspondientes al semestre 2004-2 (agosto-diciembre 2004). El principal objetivo de esta experimentación fue el que los alumnos aplicaran la técnica de puntos de función para estimar el esfuerzo de la segunda y tercera iteración en base a su propio esfuerzo realizado en iteraciones previas. El objetivo secundario fue recolectar datos empíricos sobre el esfuerzo real necesario para cada iteración y observar los resultados de la aplicación de la técnica.

4.1. Los Estudiantes y la Distribución de los Equipos

Los estudiantes fueron divididos de manera aleatoria en equipos de 11 ó 12 personas. Para la conformación de equipos se siguieron dos criterios: igual proporción de alumnos con rendimiento académico similar entre los equipos e igual proporción de hombres y mujeres. Estos criterios fueron considerados, ya que empíricamente los profesores de la carrera han comprobado que estos factores influyen en el desempeño de los proyectos seguidos en los cursos.
La tabla 2 muestra los conocimientos y experiencias de los alumnos antes de comenzar el semestre en el que se realizó esta experimentación.

Tabla 2. Conocimientos y experiencias de los estudiantes al inicio del semestre

Característica	Conocimiento y/o Experiencia
Lenguajes y Entornos de Programación	• Java, C#, Pascal, C y Prolog. • No conocían Delphi.
Bases de Datos	• Oracle 8. • No conocían MS SQL Server.
Técnicas de análisis y diseño	• Estructurado y orientado a objetos
Gestión de proyectos	• Experiencia en proyectos pequeños de programación (equipos de 3 ó 4 estudiantes). • No conocían técnicas de planificación ni estimación.

Muchos de los cursos de la carrera de Ing. Informática de la PUCP consideran proyectos como parte aplicativa de los conceptos teóricos impartidos. Los alumnos antes de tomar el curso ya tenían experiencia en realizar trabajos grupales, aunque no conocían técnicas de planificación y estimación de proyectos de software.

4.2. Características del Proyecto

Cada equipo tenía que desarrollar un sistema para una cadena de tiendas por departamentos siguiendo la metodología RUP[17]. La duración del proyecto fue de 14 semanas que es la duración del semestre académico. Antes de realizar la fase de construcción, cada equipo tuvo que finalizar el documento de especificación de requisitos (ERS) [8] con casos de uso. La arquitectura del sistema se basó en el modelo cliente/servidor de dos capas [21] y fue validado mediante la realización de un prototipo.
Todos los equipos tenían que realizar tres iteraciones de la fase de construcción. Cada iteración duró dos semanas. La cantidad de casos de uso a implementar en la segunda y tercera iteración se basaba en el esfuerzo estimado en iteraciones previas.
A cada equipo se le asignó un asistente de docencia (al que se le conoce en la PUCP con el nombre de jefe de práctica) que se encargaba de asesorar a los estudiantes en el desarrollo del proyecto. Ellos apoyaron a los estudiantes para obtener un ERS que soporte los procesos del negocio de una verdadera cadena de tiendas por departamentos. Todos los asistentes de docencia trabajan en proyectos de desarrollo de software en la industria con una experiencia promedio de siete años.
La siguiente tabla muestra el software utilizado para el desarrollo del proyecto.

Tabla 3. Software utilizado en el proyecto

Tipo	Software
Herramienta de programación	Delphi 7.0
Sistema operativo	Microsoft Windows 2000
Base de Datos	Microsoft SQL Server 2000
Herramienta de modelado	Rational Rose
Herramienta para documentar	MS Office 2003

4.3. Recolección de Datos

Semanalmente los alumnos tenían que entregar de manera individual una hoja de cálculo en MS Excel, en el que incluían las horas trabajadas en cada actividad por día. La figura siguiente muestra un ejemplo de la hoja que utilizaron todos los integrantes de los equipos.

		Semana						6	Semana						7
	Actividad	26-09-2004	27-09-2004	28-09-2004	29-09-2004	30-09-2004	01-10-2004	02-10-2004	03-10-2004	04-10-2004	05-10-2004	06-10-2004	07-10-2004	08-10-2004	09-10-2004
1	Reuniones de coordinación (internas)														
2	Reuniones de control de proyecto (con JP y/o profesor)														
3	Elaboración de informes (no incluye ningún tipo de plan)														
4	Actividades de gestión de proyectos (edt, riesgos, gantt, plan del proyecto, plan de iteración).														
5	Determinación de requisitos (lista de exigencias , especificación casos uso).														
6	Determinación de la arquitectura del sistema														
7	Preparación del plan de pruebas.														
8	Diseño de interfaz de usuario									2	2	2	2	2	2
9	Diseño (no interfaz de usuario) y programación														
10	Pruebas de software.														
11	Integración de software.														
12	Elaboración de manual de usuario														
	TOTAL	0	0	0	0	0	0	0	0	2	2	2	2	2	2
		Acum. Semanal						0	Acum. Semanal						12

Figura 2. Ejemplo de Hoja para el Llenado de Horas Trabajadas.

Es importante mencionar que se les recalcó a los alumnos que la cantidad de horas que utilizaran en el proyecto no iba a influir en la calificación final del curso, que lo importante era que cumplieran con los casos de uso que se habían comprometido a hacer para cada iteración. Esto se hizo para asegurar la honestidad en el registro del esfuerzo realizado por cada uno de los integrantes de cada equipo.

Mediante reuniones con cada equipo, se verificó que el criterio para el llenado de las hojas sea el mismo para todos. Inicialmente hubo problemas, ya que habían alumnos que registraban más horas de lo que realmente utilizaban para el proyecto.

5. RESULTADOS OBTENIDOS.

En esta sección explicarán inicialmente los resultados obtenidos previamente al estudio realizado. Luego, se mostrarán los resultados obtenidos en el semestre 2004-2.

5.1 Resultados Obtenidos Previamente a Este Estudio.

Esta técnica propuesta ha sido usada en proyectos de software en los que participa una sola persona con resultados alentadores (un ejemplo de su aplicación se encuentra documentada en una tesis de máster [15]). Los resultados obtenidos nos animaron a probar la propuesta en nuevos proyectos, incluyendo aquellos en los que participan más de una persona.

Durante el semestre 2004-1 (marzo-junio 2004) y dentro del mismo curso de la carrera de Ing. Informática, se aplicó la técnica mostrada en la sección anterior, en un proyecto de software, pero con las siguientes variantes:

- A los estudiantes no se les explicó la técnica mencionada en la sección 3 de este artículo y se les dijo que se distribuyeran la carga de trabajo usando los puntos de función sin ajustar relacionados a las transacciones.
- La planificación para saber que construir en cada iteración la hicieron en base al diagrama de precedencias.
- No se les pidió que hicieran el cálculo del esfuerzo, tal y como lo indica la técnica mencionada en la sección 3 de este artículo, sólo registraron las horas que trabajaron para el proyecto.

Se conformaron dos equipos de trabajo y cada uno de ellos tenía que desarrollar un software para una cadena de restaurantes. Cada equipo era independiente y aunque el tema era el mismo, los ERS de ambos equipos fueron diferentes entre sí. Cada equipo tenía asignado un asistente de docencia (jefe de práctica) con mucha experiencia en el tema y, que además, trabajaba hace 3 ó 4 años en una empresa de desarrollo de software.

Los integrantes de los equipos habían trabajado de manera conjunta en proyectos de asignaturas de semestres anteriores, por lo que el contexto correspondiente a las relaciones interpersonales fue la misma en todas las iteraciones. Además, se pudo observar que el único cambio que se produjo en el contexto de cada una de las iteraciones fue el conocimiento de la herramienta de programación y la experiencia en el desarrollo de aplicaciones, (factor de coste LEXP y AEXP).
A manera de ejemplo, la tabla 4, muestra los resultados obtenidos en uno de los dos equipos que trabajaron en el curso. En [16] se puede encontrar información más detallada de este estudio.

Tabla 4. Resultados para el grupo A (2004-1)

Iteración de Constr.	Factor EAF COCOMO II	Esfuerzo Real*
1	1.46	4.01
2	0.80	2.52
3	0.80	2.64

*Medido en horas-hombre/PFsA

Con la información obtenida por los dos equipos, se hizo el cálculo del índice de correlación lineal, entre el factor EAF de COCOMO II y el esfuerzo real utilizado en cada iteración. Los resultados se muestran en la tabla siguiente.

Tabla 5: Coeficiente de correlación de regresión lineal de EAF vs. esfuerzo por PFsA

Equipo	Número de Integrantes	Coeficiente Correlación
A	11	0.999
B	10	0.999

En la tabla anterior se puede observar que existe una alta correlación entre el factor EAF de COCOMO II y el esfuerzo real utilizado. Esto sirvió para utilizar factores EAF similares para la siguiente iteración.

4.2 Resultados Obtenidos en esta Experimentación.

El tema del proyecto del semestre 2004-1 fue diferente al tema propuesto en el semestre 2004-2. Esto se hizo para evitar posibles copias de lo realizado en el semestre anterior (el primero fue un sistema para una cadena de restaurantes tipo fast-food y el segundo, un sistema para una cadena de tiendas por departamentos). Aunque fueron proyectos diferentes, ambos corresponden a un tipo de sistema de información, cuyos procesos de ingreso y salida de datos son similares. Es por ello, que los resultados en el semestre 2004-1 se pueden comparar con los resultados obtenidos en el semestre 2004-2.
Debido a los resultados alentadores mostrados en la sección anterior, se decidió probar la técnica en el siguiente semestre (2004-2), con la diferencia de que los esfuerzos de las iteraciones sean calculados en base al esfuerzo real obtenido por el propio equipo en iteraciones previas, en vez de utilizar COCOMO II.
Se conformaron tres equipos de trabajo. Los datos recopilados en uno ellos fueron considerados poco confiables debido a que se produjeron algunos problemas durante desarrollo del proyecto, entre los que se incluyen discrepancias entre sus miembros.
La tabla 6 muestra los resultados del equipo A (11 participantes) y la tabla 7 del equipo B (12 participantes).

Tabla 6. Resultados para el equipo A

Iteración de Construcción	PFsA	Esfuerzo Real (horas-hombre)
1	172,66	526
2	208,22	324
3	86,12	220
TOTALES	**467,00**	**1070**

Tabla 7. Resultados para el equipo B

Iteración de Construcción	PFsA	Esfuerzo Real (horas-hombre)
1	159,13	667
2	135,87	298
3	91,00	308
TOTALES	**386,00**	**1273**

Para la fase de construcción, se indicó que la repartición de la carga de trabajo por integrante se debería realizar utilizando los PFsA obtenidos de la técnica mostrada en la sección 3. Es importante resaltar, que los resultados de la técnica en mención fueron revisados antes que los alumnos lo apliquen en su proyecto, debido a que recién estaban aprendiendo la técnica de puntos de función.

En el país no existen personas certificadas en la técnica de Puntos de Función, por lo que no se pudieron validar los resultados, aunque sí se siguieron las recomendaciones e indicaciones del manual de Puntos de Función [23]. Además, una de las personas que revisó la documentación tiene dos años de experiencia en el tema y fue alumno del máster de Ing. de Software de la Universidad Politécnica de Madrid, en donde se les enseña y les exige la aplicación de la técnica en mención para el proyecto de fin de máster.
En los resultados obtenidos en las tablas anteriores, no se han considerado las horas correspondientes a las reuniones internas de coordinación (punto 1 de la figura 2) y las reuniones con el asistente de docencia o profesor (punto 2 de la figura 2). Esto se debe a que sólo se quería considerar el trabajo efectivo realizado para la construcción del software (diseño, programación y pruebas).
Para los resultados mostrados en las tablas 7 y 8 de esta sección, se ha utilizado la magnitud de error relativo (MRE), la cual se define como sigue (*y*=esfuerzo real, $\hat{y}$ =esfuerzo estimado) [6]

$$MRE = \frac{\left| y - \hat{y} \right|}{y}$$

Para el esfuerzo estimado de cada iteración se ha utilizado la siguiente fórmula (***esf_estimado(i)***=esfuerzo estimado para la iteración "i", ***EAF(i)***= factor EAF para la iteración *"i"* y ***esf_real(j)***=esfuerzo real de la iteración *"j"*)

$$esf\_estimado(i) = \frac{EAF(i)}{i-1} \times \sum_{j=1}^{i-1} \frac{esf\_real(j)}{EAF(j)}$$

Tabla 7. Resultados para el equipo A

Iter. Constr.	Factor EAF	Esfuerzo Real*	Esfuerzo Estimado*	MRE
1	1,46	3,05	-	-
2	0,80	1,56	1,67	7,28%
3	0,80	2,55	1,61	36,87%

*Medido en horas-hombre/PFsA

Tabla 8. Resultados para el equipo B

Iter. Constr.	Factor EAF	Esfuerzo Real*	Esfuerzo Estimado*	MRE
1	1,46	4,19	-	-
2	0,80	2,19	2,30	4,72%
3	0,80	3,38	2,25	33,67%

De manera similar a lo ocurrido en el semestre 2004-1 (ver tabla 4), se pudo observar que los cambios que se produjeron en el contexto de cada una de las iteraciones fue el conocimiento de la herramienta de programación y la experiencia en el desarrollo de aplicaciones, (factor de coste LEXP y AEXP). Es por eso que los factores EAF de las tablas 7 y 8 son los mismos que al de la tabla 6.
De los resultados obtenidos, se puede observar que el MRE de la segunda iteración es menor al 8% en ambos equipos, esto quiere decir que la técnica utilizada fue bastante acertada para esta iteración En cambio, en la tercera iteración, se puede observar que el MRE es mayor al 33%, lo que podría suponer que la técnica falló para esta iteración. Inicialmente, se pensó que hubo un error en la recopilación de datos, pero luego de conversar con los alumnos hubo un detalle que no se tomó en cuenta. Para la última entrega, los alumnos tenían que presentar su proyecto a todos sus compañeros del curso, a todos los asistentes de docencia y al profesor, por lo que la presión de presentar un buen trabajo fue muchísimo mayor que en las otras iteraciones. Esto provocó que los equipos hicieran pruebas más intensivas del software e incluso modificaron partes que ya habían sido probadas y que funcionaban de manera correcta.
Para este problema se puede considerar lo que propone COCOMO II [4] que incluye un modelo para estimar el porcentaje de tamaño de software que se modificará de las iteraciones anteriores. A diferencia de la segunda iteración en la que no hubo problemas en el cálculo, ya que los alumnos no modificaron la funcionalidad hecha en la primera iteración.
En la siguiente tabla se muestra el porcentaje de modificación realizada de los puntos de función de la primera y segunda iteración, en la tercera iteración

Tabla 9. Porcentaje de uso de PFsA de 1ra y 2da iteración en la 3ra iteración.

Equipo	PFsA 1ra & 2da iteración*	PFsA 3ra iteración*	Total PFsA Usadas 3ra iteración**	% de Uso de PFsA de la 1ra y 2da Iteración
A	380.88	86.12	137.79	13.21%
B	295.00	91.00	138.56	15.66%

* PFsA tomado de la técnica propuesta
** PFsA calculados en base al esfuerzo estimado para la tercera iteración y el esfuerzo real realizado en esa iteración.

En la tabla anterior se puede observar que el porcentaje de modificación realizada de los PFsA de la primera y segunda iteración, en la tercera fue de aproximadamente 14%. Este dato será utilizado en futuras experiencias con alumnos.

6. DISCUSIÓN DE LOS RESULTADOS OBTENIDOS

Aunque la información obtenida corresponde a dos proyectos y los resultados obtenidos son alentadores para la técnica propuesta, aún no se puede afirmar que la técnica sea realmente confiable para otros contextos. Algunos de los comentarios de los alumnos con respecto a la técnica fueron los siguientes:

- La técnica les sirvió para medir el tamaño de la parte del software que tenían que implementar cada integrante para una iteración.
- El esfuerzo registrado en la primera iteración fue muy útil para calcular el esfuerzo de la segunda iteración.

También, se puede observar (ver la tabla 10) que el esfuerzo utilizado de ambos equipos es diferente. Por ejemplo, en la segunda iteración, el equipo A requirió de 1.56 horas-hombre/PFsA, en cambio el equipo B requirió 2.19 horas-hombre/PFsA. Algo similar ocurre para las otras iteraciones.

Tabla 10. Esfuerzo real para cada iteración por equipo

Equipo	Esfuerzo Real 1ra Iteración*	Esfuerzo Real 2da Iteración	Esfuerzo Real 3ra Iteración*
A (11 alumnos)	3.05	1.56	2.55
B (12 alumnos)	4.19	2.19	3.88

*Medido en horas-hombre/PFsA

La variación que se muestra en la tabla anterior se debe a muchos factores, alguno de ellos son: la cantidad de alumnos por equipo y la capacidad de trabajo en equipo, siendo algunos de ellos difíciles de cuantificar. Lo que sí se pudo comprobar es que la información del esfuerzo real utilizado es muy útil para la estimación del esfuerzo del equipo a quien le pertenecen esos datos.

En cuanto a los resultados obtenidos en la tercera iteración, el MRE fue aproximadamente al 35% para todos los proyectos y equipos (tablas 7 y 8), como se comentó esto se debía a los cambios que realizaron los primeros incrementos, debido a la presión producida por la presentación final. En la tabla 9 se puede mostrar que el porcentaje de modificación realizada de la primera y segunda iteración, en la tercera fue de aproximadamente 14%. Para ambas técnicas el porcentaje es muy similar. Este dato será utilizado en futuras experiencias con alumnos.

Es difícil hacer generalizaciones de los resultados obtenidos para otros contextos que no sea el académico. La ventaja de realizar experimentaciones con alumnos es el que se puedan controlar algunos factores o variables que no se pueden controlar en proyectos en la industria. Algunos de ellos son los siguientes:

- Conocimiento y experiencia de los miembros de los equipos de desarrollo
  En proyectos de la industria es muy difícil encontrar que la experiencia de todos los miembros sea la misma. En algunos casos, los miembros no tienen los conocimientos necesarios sobre desarrollo orientado a objetos.
- Rotación de personal
  En la industria es bastante frecuente el cambio de los integrantes de un proyecto de software por diversos factores, entre ellos, la asignación a otros proyectos. Para la experimentación, este factor no se produjo.
- Cambio de requisitos
  El cambio de requisitos es algo que ocurre frecuentemente en la industria. Este factor se puede controlar en un proyecto académico.
- Secuencia de implementación de los requisitos
  En esta experimentación se ha realizado la implementación de los requisitos siguiendo el diagrama de precedencias, lo cual a veces no se puede hacer realizar en la industria. Esto se debe a que la precedencia de construcción de los requisitos depende de los requerimientos de los usuarios y de los clientes.

Los resultados obtenidos se podrían aplicar en la industria cuando el equipo de trabajo cuente mucha experiencia en el desarrollo de aplicaciones y tenga que afrontar un proyecto en el que se tenga que utilizar una nueva herramienta y lenguaje de programación.

7. CONCLUSIONES Y TRABAJO FUTURO

La mayoría de las aproximaciones actuales para estimar proyectos, aún definiéndose como independientes de la tecnología y modelos de ciclo de vida, tienen un carácter fuertemente influido por los ciclos de vida en cascada. A pesar de que son válidas para otras aproximaciones, como los ciclos de vida iterativos-incrementales, por ejemplo, en general no ofrecen ninguna guía para acometer dicha adaptación.

El presente trabajo muestra una técnica que se basa en Puntos de Función para la estimación del esfuerzo en proyectos que utilizan un ciclo de vida iterativo-incremental. Además, se muestran los resultados obtenidos en proyectos de software realizados por estudiantes. La ventaja de utilizar alumnos es que se pueden controlar factores que podrían afectar el estudio, como por ejemplo, conocimiento y habilidades de los participantes y contexto de cada iteración.

Los resultados obtenidos son bastante alentadores, ya que la técnica propuesta da un error relativo entre el esfuerzo estimado y esfuerzo real menor al 10%. Los equipos utilizaron los datos de las iteraciones previas para hacer el cálculo del esfuerzo estimado para las siguientes iteraciones. Estos resultados no son concluyentes, ya que se necesitan realizar más pruebas, no sólo con alumnos sino también en la industria.

El trabajo futuro que está relacionado a este artículo es el siguiente:

- Adaptar la técnica propuesta de manera que los ficheros sean reemplazados por diagramas de clases, para que se adapte al enfoque orientado a objetos.
- Realizar experiencias en las que no se puede seguir la implementación con el orden sugerido por el diagrama de precedencias, ya que lo que muchas veces ocurre que la prioridad de implementación requerida en proyectos en la industria es muy diferente.
- Adaptar la técnica propuesta a otras técnicas como COSMIC-FFP y Puntos de Casos de Uso.
- Realizar experiencias con los alumnos para comparar la efectividad de otras técnicas, como COSMIC-FFP y Puntos de Casos de Uso, para estimar el esfuerzo de las iteraciones en proyectos de software.

Agradecimientos

Agradezco a la Dra. Ana M. Moreno, profesora de la Universidad Politécnica de Madrid, por sus comentarios a la versión inicial de este artículo. Asimismo, mi reconocimiento a todos los alumnos que participaron en esta experimentación y cuyos nombres de equipo fueron: CSM, AISBER y 11-Son-Suficientes.

Referencias

[1] Albrecht, A. J. *Measuring Application Development Productivity*, IBM Applications Development Symposium, Monterey, CA, USA, 1979.

[2] Bittner, K., *Use Case Modeling*, Addison-Wesley, USA, 2003.

[3] Bittner, K., "Why Use Cases Are Not Functions", http://www.therationaledge.com, USA, 2000.

[4] Boehm, B., et al. *Software cost estimation with COCOMO II,* Prentice-Hall, USA, 2000.

[5] Carver, J., Jaccheri, L., Morasca, S., *Issues in Using Empirical Studies in Software Engineering Education*, Proceedings METRICS'03, IEEE Computer Society, USA, 2003.

[6] Conte SD, HE Dunsmore, and VY Shen, *Software Engineering Metrics and Model*s, Benjamin-Cummings, Menlo Park CA, 1986.

[7] Fetcke, T., Bran, A., Nguyen, T., Mapping the OO-Jacobson Approach into Function Point Analysis. Proceedings of TOOLS-23'97, IEEE, USA, 1997.

[8] IEEE Computer Society, IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications, The Institute of Electrical and Electronics Engineers, USA, 1998.

[9] Jacobson, I., Object-Oriented Software Engineering. A Use Case Driven Approach, Addison-Wesley, USA, 1992.

[10] Karner, G. *Metrics for Objectory*. *Diploma thesis*, University of Linköping, Sweden. No. LiTH-IDA-Ex-9344:21. December 1993.

[11] Longstreet, D.,Use Case and Function Points, http://www.softwaremetrics.com/, Longstreet Consulting Inc, USA, 2001.

[12] Object Management Group, OMG Unified Modeling Language, http://www.uml.org, USA, 1999.

[13] Pow-Sang, J., Imbert R., Estimación y Planificación de Proyectos Software con Ciclo de Vida Iterativo-Incremental y empleo de Casos de Uso, Proceedings IDEAS 2004, Arequipa-Perú, 2004.

[14] Pow-Sang, J., La Especificación de Requisitos con Casos de Uso: Buenas y Malas Prácticas, II Simposio Internacional de Sistemas de Información e Ing. de Software en la Sociedad del Conocimiento-SISOFT 2003, Pontificia Universidad Católica del Perú, Lima-Perú, 2003.

[15] Pow-Sang, J., GESPROMET, Sistema para la Gestión de Proyectos de Software Utilizando MÉTRICA Versión 3. Tesis de Máster en Ingeniería del Software, Universidad Politécnica de Madrid, España, 2002.

[16] Pow-Sang, J., *Estudio Comparativo de Técnicas para la Estimación del Esfuerzo de las Iteraciones de Proyectos Software*, Proceedings JIISIC'04, Madrid-España, 2004.

[17] Rational Software, Rational Unified Process version 2001A.04.00.13, USA, 2001.

[18] Royce, W. W., Managing the Development of Large Software Systems: Concepts and Techniques. Proceedings WESCON, 1970.

[19] Rumbaugh, I. Jacobson, and G. Booch, Unified ModellingLanguage Reference Manual, Addison Wesley, 1997.

[20] Rosenberg, D., Scott, K., Use Case Driven Object Modeling with UML, Addison-Wesley, Massachusets, USA, 1999.

[21] Sadoski, D., Two Tier Software Architectures, http://www.sei.cmu.edu/str/descriptions/twotier.html, SEI, USA, 2004.

[22]Schneider, G. and Winters, J. *Applying Use Cases – A Practical Guide*, 2nd Edition. Addison-Wesley. USA, 2001.

[23] The International Function Point User Group (IFPUG), Function Point Counting Practices Manual-Release 4.1, USA, 1999.