Una Propuesta para la Estimación del Esfuerzo de las Iteraciones Utilizando Puntos de Casos de Uso

José Antonio Pow-Sang Portillo
Pontificia Universidad Católica del Perú, Departamento 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 a technique to estimate and planning software projects that use iterative-incremental lifecycles. This technique is based on Use Case Points and COCOMO II. The results of the application of this technique in software development projects with students are also included in this paper.

Keywords: Effort estimation, Iterative-incremental lifecycle, 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.

El presente artículo presenta una propuesta para estimar y planificar proyectos software cuyo ciclo de vida sea el iterativo-incremental utilizando Puntos de Caso de Uso y COCOMO II. Además, se incluyen los resultados obtenidos al aplicar la técnica en proyectos desarrollados por estudiantes.

Palabras claves: Estimación del esfuerzo, Ciclo de vida iterativo-incremental, Casos de uso

1. INTRODUCCIÓN

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 [20], Puntos de Casos de Uso [8] y COCOMO II [3]. Aún así, estas técnicas fueron concebidas para su aplicación en sistemas basados en un ciclo de vida clásico o en cascada de Royce [18], y aún es difícil emplearlas en desarrollos con ciclos de vida iterativo-incrementales, tan en boga en los últimos años.

Teniendo en cuenta la problemática planteada, el presente artículo propone una técnica para estimar y planificar las iteraciones en proyectos orientados a objetos, basada en los casos de uso de los mismos, la técnica de Puntos de Casos de Uso y el método de COCOMO II.

Este documento se ha estructurado de la siguiente manera: la sección 2 muestra un breve resumen de las técnicas de estimación; la sección 3, plantea una propuesta para la estimación y planificación de las iteraciones; la sección 4, muestra los resultados obtenidos al aplicar la técnica en un proyecto de software realizado por estudiantes; y, finalmente, se presentan las conclusiones obtenidas de este trabajo.

2. LAS TÉCNICAS DE ESTIMACIÓN

Esta sección muestra un breve resumen de las técnicas de Puntos de Caso de Uso y COCOMO II. Además, se da una visión general del trabajo que se ha encontrado con respecto al tema.

Los Puntos de Caso de Uso (PCU)

Esta técnica, conocida en inglés como Use Case Points, fue desarrollada por Gustav Karner [9] y es una extensión de Puntos de Función [19]. Los pasos que propone esta técnica son los siguientes:

1. Los actores del modelo de casos de uso son categorizados como simples, medios o complejos. Un actor simple representa otro sistema con un API definido, un actor medio es otro sistema que interactúa a través de un protocolo como TCP/IP y un actor complejo es una persona que interactúa con el sistema a través de una interfaz gráfica de usuario o una página Web.
2. Los casos de uso son categorizados como simples, medios o complejos, dependiendo en el número de transacciones que se encuentren especificadas en cada uno de ellos. Los casos de uso incluidos y extendidos no son considerados. Un caso de uso simple tiene menos de 4 transacciones, uno medio tiene de 4 a 7 transacciones y uno complejo, más de 7.
3. Cada categoría de actor y de casos de uso se cuantifica en Puntos de Caso de Uso sin Ajustar (PCUsA). Luego, se obtiene la suma total de PCUsA debido a todos los actores y casos de uso de todo el sistema.
4. Los PCUsA son multiplicados por un factor de ajuste, que se calcula en base a factores técnicos y de entorno del proyecto. Con este procedimiento se obtienen los Puntos de Casos de Uso Ajustados. La técnica indica que se debe considerar un esfuerzo de 20 horas-hombre por cada Punto de Caso de Uso sin Ajustar obtenido.

COCOMO II

COCOMO II, propuesto y desarrollado por Barry Boehm [3], 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

Cost Drivers	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

Los PCU y los Ciclos de Vida Iterativos-Incrementales.

Actualmente, existe poca información y experimentación en el uso de los PCU y trabajo con incrementos. Una de las propuestas más recientes es la de Mohagheghi [10] en la que muestra los resultados de una adaptación de la técnica de PCU para trabajar con incrementos, para ello usa el modelo de COCOMO II para ajustar los incrementos en base a la reutilización y modificación de lo construido en iteraciones anteriores. La propuesta ha sido probada en proyectos a gran escala.

Uno de los problemas encontrados al querer distribuir los PCU en cada iteración, y que no ha sido mencionado como hacerlo en [10], es la distribución de los PCU correspondientes a los actores entre cada una de las iteraciones. En el caso que todos los actores tengan la misma complejidad, puede ser que no sea necesario considerarlos para el conteo de cada iteración. Además, el peso de ellos es muy poco en comparación a los PCU obtenidos por cada caso de uso.

3. PROPUESTA DE ESTIMACIÓN Y PLANIFICACIÓN UTILIZANDO PUNTOS DE CASOS DE USO

La propuesta para determinar la estimación y planificación del esfuerzo de un desarrollo que se apoya en un ciclo de vida iterativo-incremental comprende dos pasos principales: en primer lugar, determinar los casos de uso a realizar por iteración y, posteriormente, estimar el esfuerzo para cada iteración. Esta técnica es una adaptación a la que se incluye en [12]. A continuación se detallan las actividades que se siguen en cada uno de los pasos mencionados.

Paso 1: Determinar los Casos de Uso a Realizar por Iteración

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 revisar [13][2][8] y para el documento de especificación de requisitos de software, [7][16]).

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 Rosenberg [18] 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.

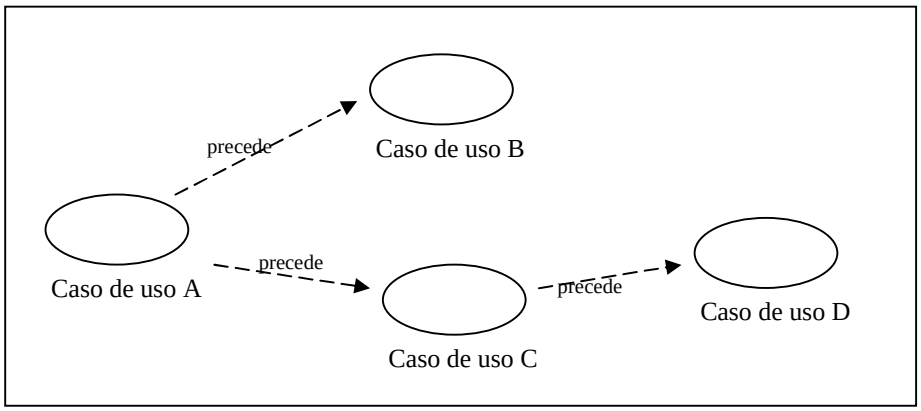

Figura 1. Ejemplo de Diagrama de Precedencias

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.

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.

Paso 2: Estimación del Esfuerzo para cada Iteración

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 Casos de Uso y COCOMO II.

Paso 2.a: Determinar los PCUsA para cada Caso de Uso

Inicialmente se realiza la estimación del tamaño de cada caso de uso utilizando la técnica PCU. Para este caso, no se han considerado los PCUsA correspondientes a los actores, debido a que está muy relacionado a la complejidad de la interfaz de usuario y lo que se quiere considerar es el tamaño del software a realizar sin importar detalles correspondientes a implementación.

Una variante a lo anterior, es incluir el peso por los ficheros que se utilizarán en cada caso de uso. Por ello, y de una manera similar a la técnica de Puntos de Función [19], se determinan los ficheros y las complejidades de cada uno de ellos. El peso en PCUsA para cada fichero es el siguiente:

Tabla 1. Peso en PCUsA debido a los ILFs

Tipo de ILF	Peso
Bajo	10
Medio	15
Alto	20

Tabla 2. Peso en PCUsA debido a los EIFs

Tipo de ILF	Peso
Bajo	5
Medio	10
Alto	15

Posteriormente, se determinan los PCUsA sin ajustar correspondientes a un fichero lógico interno o externo para cada caso de uso. Para ello, se utiliza la siguiente fórmula (*PCUsA_F* = puntos de casos de uso 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)

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

Con los resultados obtenidos con la fórmula anterior y sabiendo qué caso de uso se desarrollará en cada iteración, se determinan los PCUsA debido a los ficheros por iteración. A esto se le añadirán los PCUsA correspondientes a los casos de uso. Para ello, se utiliza la fórmula que se muestra a continuación (*i*= iteración específica, *TPCUsA(i)*= total de puntos de caso de uso sin ajustar para una iteración "*i*", *PCUsA_F(j)* = puntos de caso de uso sin ajustar para un caso de uso "*j*" debido a ILF/EIFs, *PCUsA_CU(j)* = puntos de caso de uso sin ajustar para un caso de uso "*j*" debido a la complejidad del caso de uso (EI,EO,EQ) y *j*= caso de uso a implementar en una iteración "*i*").

$$TPCUsA(i) = \sum_{j=1}^{n} \left[PCUsA\_F(j) \right] + \sum_{j=1}^{n} \left[PCUsA\_CU(j) \right]$$

Cabe resaltar que existen algunas propuestas para el uso del diagrama de clases de análisis en reemplazo de los ficheros de la técnica de puntos de función [1][6], este tema será considerado en otros artículos.

Paso 2.b Determinación del esfuerzo utilizando COCOMO II

Para realizar este paso, en primer lugar es preciso estimar cada uno de los *drivers* de coste propuestos por el método COCOMO II (RELY, DATA, etc), correspondientes a cada iteración. Con esto, se podrá calcular el factor *EAF* el cual representará cuantitativamente el contexto de la iteración, la cual puede cambiar de una iteración a otra (conocimiento de la plataforma de desarrollo, integración del equipo de desarrollo, etc).

A partir del factor *EAF*, los PCUsA correspondientes a cada iteración y a información histórica, se podrá estimar el esfuerzo que se necesita para la primera iteración. Para las siguientes iteraciones se puede utilizar los esfuerzos reales que se requirieron para completar la primera iteración para estimar el esfuerzo requerido para la segunda iteración.

4. RESULTADOS.

Esta sección muestra los resultados obtenidos al aplicar la técnica con los datos recopilados de proyectos con alumnos. Los estudiantes utilizaron una técnica similar a la mostrada en este artículo, pero con puntos de función para estimar su proyecto (los resultados de esta experimentación se muestran de manera detallada en [15]). Luego de finalizar los proyectos y con la información recolectada, se pudo realizar los cálculos, de manera similar a lo realizado con Puntos de Función, pero utilizando Puntos de Casos de Uso.

Esta sección se divide en dos acápites: el primero muestra la descripción de los proyectos realizados por los alumnos y el segundo, los resultados obtenidos al aplicar la técnica.

La Descripción de los Proyectos.

Los proyectos fueron realizados por estudiantes de cuarto año de la carrera de Ing. Informática de la Pontificia Universidad Católica del Perú. Esta experiencia se desarrolló en una asignatura obligatoria del área de Ingeniería 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.

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 3 muestra los conocimientos y experiencias de los alumnos antes de comenzar el semestre.

Tabla 3. 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 (basic). • No conocían Delphi.
Bases de Datos	• Oracle 8. • No conocía 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 conocía técnicas de planificación ni estimación.

Cada equipo tenía que desarrollar un sistema para una cadena de tiendas por departamentos siguiendo la metodología RUP[16]. 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) [7] con casos de uso. La arquitectura del sistema se basó en el modelo cliente/servidor de dos capas 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.

La siguiente tabla muestra los aspectos técnicos para el desarrollo del proyecto.

Tabla 4. Aspectos Técnicos del Proyecto

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

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.

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.

Resultados Obtenidos al Aplicar la Técnica

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) [5]

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

Para el esfuerzo estimado de cada iteración se ha utilizado la siguiente fórmula (***esf_estimadot(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"*)

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

De manera similar a una experimentación realizada en el semestre 2004-1 (para más detalles de la experimentación puede ver [14]), 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 son 1,46 para la primera iteración y 0,8 para la segunda iteración.

Tabla 5. Resultados para el equipo A sin Ficheros

Iter. Constr.	Factor EAF	Esfuerzo Real*	Esfuerzo Estimado*	MRE
1	1,46	2.62	-	-
2	0,80	1.45	1,43	1,40%

*Medido en horas-hombre/UCPsA

Tabla 6. Resultados para el equipo A con Ficheros

Iter. Constr.	Factor EAF	Esfuerzo Real*	Esfuerzo Estimado*	MRE
1	1,46	1,86	-	-
2	0,80	1,01	1,02	0,67%

Tabla 7. Resultados para el equipo B sin Ficheros

Iter. Constr.	Factor EAF	Esfuerzo Real*	Esfuerzo Estimado*	MRE
1	1,46	2.63	-	-
2	0,80	1.58	1,44	8,82%

*Medido en horas-hombre/UCPsA

Tabla 8. Resultados para el equipo B con Ficheros

Iter. Constr.	Factor EAF	Esfuerzo Real*	Esfuerzo Estimado*	MRE
1	1,46	1,65	-	-
2	0,80	0,84	0,90	8,14%

De los resultados obtenidos, se puede observar que el MRE estimado para la segunda iteración es menor al 10% en ambos equipos. Además, se puede que para la segunda iteración el MRE es más bajo cuando se utilizan los PCUsA de ficheros (tablas 6 y 8) que cuando no se usan (tablas 6 y 7). En base a estos resultados, se podría inferir que el uso de PCUsA de ficheros podría mejorar la estimación del esfuerzo.

Los resultados mostrados, aunque son alentadores, no son definitivos, ya que se necesita realizar mayor experimentación al respecto.

5. 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 basada en puntos de caso de uso, y los resultados al ser aplicados a la información obtenida en proyectos de software realizados por alumnos. 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 las técnicas propuestas dan un error relativo entre el esfuerzo estimado y esfuerzo real menor al 10%. Los equipos utilizan 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 la propuesta es la siguiente:

- Adaptar la técnica propuesta para Puntos de Casos de Uso y Puntos de Función de manera que los ficheros sean reemplazados por diagramas de clases, para que se adapte completamente 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 por la industria es muy diferente.
- Adaptar la técnica propuesta a otras técnicas como COSMIC-FFP.
- Realizar experiencias con los alumnos utilizando ambas técnicas para estimar el esfuerzo de las iteraciones.

REFERENCIAS

1. Abrahao, S., Poels, G, Pastor, O., Comparative Evaluation of Functional Size Measurement Methods: An Experimental Análisis. Working paper 2004/234 at Faculty of Economics and Business Administration- Ghent University, Belgium, February 2004.
2. Bittner, K., Use Case Modeling, Addison-Wesley, USA, 2003.
3. Boehm, B., et al. Software cost estimation with COCOMO II, Prentice-Hall, USA, 2000.
4. Carver, J., Jaccheri, L., Morasca, S., Issues in Using Empirical Studies in Software Engineering Education, Proceedings METRICS'03, IEEE Computer Society, USA, 2003.
5. Conte SD, HE Dunsmore, and VY Shen, Software Engineering Metrics and Models, Benjamin-Cummings, Menlo Park CA, 1986.
6. Fetcke, T., Bran, A., Nguyen, T., Mapping the OO-Jacobson Approach into Function Point Analysis. Proceedings of TOOLS-23'97, IEEE, USA, 1997.
7. IEEE Computer Society, IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications, The Institute of Electrical and Electronics Engineers, USA, 1998.
8. Jacobson, I., Object-Oriented Software Engineering. A Use Case Driven Approach, Addison-Wesley, USA, 1992.
9. Karner, G. Metrics for Objectory. Diploma thesis, University of Linköping, Sweden. December 1993.
10. Mohagheghi, P., Anda, B., Conradi, R., Effort Estimation of Use Cases for Incremental Large-Scale Software Development, Proceedings ICSE'05, ACM, USA.
11. Object Management Group, OMG Unified Modeling Language, http://www.uml.org, USA, 1999.
12. 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.
13. 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.
14. Pow-Sang, J., Estudio Comparativo de Técnicas para la Estimación del Esfuerzo de las Iteraciones de Proyectos Software, Proceedings JIISIC'04, Madrid-España, 2004.
15. Pow-Sang, J., Una Experiencia con Estudiantes para la Estimación del Esfuerzo de cada Iteración en Proyectos de Software, a ser publicado en el CLEI 2005, Cali-Colombia, 2005.
16. Rational Software, Rational Unified Process version 2001A.04.00.13, USA, 2001.
17. Royce, W. W., Managing the Development of Large Software Systems: Concepts and Techniques. Proceedings WESCON, 1970.
18. Rosenberg, D., Scott, K., Use Case Driven Object Modeling with UML, Addison-Wesley, Massachusets, USA, 1999.
19. The International Function Point User Group (IFPUG), Function Point Counting Practices Manual-Release 4.1, USA, 1999.