Estimación de proyectos de software pequeños basada en el juicio de expertos: un caso de estudio

Gabriela Robiolo[1] y Silvana Santos[2]

[1] Universidad Austral, Mariano Acosta 1611, B1629WWA Pilar, Buenos Aires, Argentina.
grobiolo@austral.edu.ar

[2] Universidad Nacional de La Plata, Calle 50 y 20, 1900 La Plata, Buenos Aires, Argentina
silvanasantos@gmail.com

Resumen. El Juicio de Expertos es el método de estimación de proyectos de software más comúnmente utilizado entre los desarrolladores. En la actualidad existen pocos estudios empíricos relativos a la estimación de esfuerzo de proyectos de software. En este trabajo presentamos un caso de estudio desarrollado en una empresa internacional donde surge la necesidad de mejorar la estimación de proyectos pequeños de software. Con este motivo se formaliza la definición de los requerimientos y se aplica un método de estimación basado en el juicio de expertos. El resultado obtenido puso en evidencia la necesidad de considerar una contingencia del 39% para evitar errores de más del 10% en las estimaciones.

1 Introducción

En el contexto de la ingeniería de software persiste la preocupación por los fracasos de proyectos de software. Charette [1] puntualiza las siguientes causas: objetivos de proyectos no realistas o inarticulados; estimaciones de los recursos necesarios inexactas, requerimientos mal definidos; estado del proyecto mal reportado; riesgos no administrados; comunicación ineficiente entre clientes, desarrolladores y usuarios; inmadurez en las tecnologías utilizadas; inhabilidad para controlar la complejidad de los proyectos; descuido en las prácticas de desarrollo de software; administración de proyectos insuficiente y la deficiente armonización de las políticas de los actores involucrados y las presiones comerciales con los objetivos del proyecto.

En las últimas décadas el aumento del tamaño y la complejidad de los proyectos de software han aumentado el valor económico de los fracasos de proyectos. Además, la presencia de realidades informatizadas en todos los ámbitos de la vida social lleva a impactar en pequeña y gran escala a numerosas personas e instituciones.

Dentro de las causas de fracasos de proyectos de software enunciadas anteriormente, que se entienden muy cercanas a la realidad, en este trabajo interesa poner foco en las estimaciones de recursos, focalizando el estudio en el costo de desarrollo medido principalmente en horas persona.

Jørgensen [2] indica que en su revisión sistemática de estimación experta no se han logrado encontrar estudios que reportasen que la mayoría de las estimaciones fueron basadas en métodos de estimación formales. Hay evidencia disponible [2] que sugiere que la precisión en la estimación no mejora con el uso formal de modelos de estimación. A pesar de esto, las investigaciones sobre estimación experta es escaza.

Jørgensen y Shepperd [3] afirman que existen pocos investigadores interesados en este tema y que además es necesario definir un marco adecuado para el desarrollo de trabajos de calidad. Los autores proponen realizar las siguientes mejoras: estudios que den soporte al método de estimación basadas en el juicio de expertos en lugar de reemplazarlos por otros métodos de estimación. Siendo el juicio de expertos el método más usado en la industria del software, sería conveniente potenciar este juicio usando los métodos formales de estimación. Además, la aplicación de métodos de estimaciones de costos aplicados en situaciones reales. Existen pocos estudios donde los métodos de estimación son evaluados en situaciones reales, puesto que la mayoría se han aplicado en contextos no reales de laboratorios.

Las consideraciones anteriores nos han llevado a interesarnos por el método de estimación de esfuerzo de proyectos de software basada en el juicio de expertos, analizando un caso de estudio en el área de desarrollo de software de una empresa multinacional que da soporte a sus actividades comerciales.

Particularmente, el objetivo de este trabajo busca mejorar el método de estimación de esfuerzo de pequeños proyectos de software (PPS) basado en el juicio de expertos, aplicado en una empresa con la finalidad de reducir el error en las estimaciones de esfuerzo medido en horas-persona y aumentar la satisfacción del cliente. Concretamente, la gerencia de dicha empresa solicitó al área de desarrollo trabajar con una mayor precisión en las estimaciones para hacer un uso más eficiente de los recursos. Por lo tanto se plantea la siguiente pregunta de investigación:

¿Es posible mejorar la estimación de esfuerzo medido en horas persona de los PPS?

Organizamos el artículo de la siguiente forma: en primer lugar se describen los métodos de estimación que se estudiaron como marco teórico, a continuación se describe el problema que originó el cambio en el método de estimación de proyectos en un área de desarrollo de software de una empresa multinacional y la solución implementada. Finalmente se realiza una discusión sobre los resultados obtenidos y las lecciones aprendidas, sintetizadas en las conclusiones.

2 Métodos de estimación basados en la experiencia de expertos

Se seleccionaron los métodos de estimación basados en el Juicio de Expertos más usados en la industria del software en la actualidad [2][3] junto con Wideband Delphi [4] y Planing Poker [5] por ser usado en las metodologías de desarrollo de software ágiles. Motivo esta selección el hecho que el caso de estudio a analizar se desarrolló en un contexto industrial. Una explicación más detallada la puede encontrar en [6].

Juicio de Expertos. Esta estrategia de estimación se lleva a cabo por una persona reconocida como un experto en la tarea, y una parte significativa del proceso de estimación se basa en un proceso de razonamiento no explícita y no recuperable, es decir, "la intuición" [2]. Según un estudio realizado por [3], el método de estimación dominante es Juicio de Expertos. Los autores sostienen que no hay suficiente evidencia disponible que sugiera que las estimaciones mejoran con el uso de métodos formales de estimación. Es el método de estimación dominante, más ampliamente usado no solo en el área de desarrollo de software sino también en otras áreas tales como negocios, salud, educación, etc. [2].

A continuación se presentan los 12 principios de estimación expertas validados empíricamente por [2]. Los primeros seis principios favorecen la reducción de la influencia humana y las influencias de las circunstancias, los principios 7-10 dan soporte al proceso de estimación y los dos últimos están orientados a proveer feedback y oportunidades de entrenamiento:

1) Evaluar la precisión en las estimaciones pero a la vez evitar la alta presión de evaluación.
2) Evitar metas de estimación en conflicto
3) Pedir a los estimadores que justifiquen y critiquen sus estimaciones
4) Evitar información de estimación irrelevante y poco confiable
5) Usar datos documentados de tareas de desarrollo previas
6) Buscar estimadores expertos con conocimientos relevantes del dominio y buenos antecedentes de estimaciones.
7) Estimar usando la estrategia de descomposición Top-Down y la de descomposición aditiva Bottom-Up independientemente.
8) Usar checklists de estimaciones
9) Combinar estimaciones de diferentes expertos y estrategias de estimación
10) Evaluar la incertidumbre en las estimaciones
11) Proveer feedback sobre la precisión en las estimaciones y las relaciones entre tareas
12) Proveer oportunidades de entrenamiento en estimaciones

Ejemplos de casos de aplicación: Jet Propulsion Laboratory [7]; compañías holandesas [8], una compañía de Telecom [11]; otras compañías de desarrollo de software [9].

Wideband Delphi. Barry Boehm creó Wideband Delphi y fue popularizado en el libro de Boehm “Software Engineering Economics” [10]. Es una técnica de estimación de esfuerzo basada en el consenso logrado en un grupo de expertos. Deriva de la técnica Delphi [4]. Fue llamada “Wideband” porque incluye mucha más interacción y comunicación entre los participantes distinto de la técnica Delphi original donde se evitan las discusiones de grupo [10][12].

Boehm [10] detalla los pasos del método, donde se destaca: la presencia de un coordinador, estimaciones de expertos preparadas individualmente y discutidas en grupo y votadas anónimamente.

Esta técnica fue usada para estimar la introducción de defectos de software y las tasas de extracción durante las diferentes fases del ciclo de vida de desarrollo de

software [12]. También Fue aplicado para estimar el esfuerzo que iba a demandar llevar a una organización a alcanzar el Nivel 2 de CMM [13].

Analogías. Según [14], el principio de base es caracterizar proyectos en términos de características como por ejemplo: el número de interfaces, el método de desarrollo o el tamaño de los requerimientos funcionales. Existe una base de proyectos terminados que se usa para buscar aquellos que más se asemejen al proyecto que se quiere estimar.

Según Shepperd et al. [14], la idea de usar analogías como base para estimar el esfuerzo en proyectos de software no es algo nuevo [10] sugirió el uso informal de analogías como una técnica posible hace aproximadamente 30 años atrás. La idea fue reiterada por Cowderoy y Jenkins en 1988 pero nuevamente, sin un mecanismo formal de selección de analogías. El próximo desarrollo fue realizado por Vicinanza et al. [15][16][17] quien sugirió que los desarrollos de la comunidad de la máquina de aprendizaje en forma de Case-Based Reasoning: podrían ser útilmente adaptados para ayudar a obtener mejores predicciones. Existe una herramienta ANGEL que soporta el método CBR [18].

Atkinson y Shepperd [19] realizaron un estudio sobre 21 proyectos reales donde demostraron que la aplicación de Analogías es como mínimo tan efectivo como Juicio de Expertos y/o Modelos Algorítmicos tradicionales derivados de Análisis de Regresión.

Planning Poker. Es una técnica sencilla de estimación de esfuerzo de desarrollo de software basada en el consenso al que llega el equipo de expertos que está estimando. Es un enfoque que combina opiniones de expertos, analogías y desagregación que resulta en estimaciones rápidas y confiables [20]. Utilizada mayormente en el desarrollo ágil de software, en particular en Extreme Programming [21].

Este método fue descrito por primera vez por James Greening [22] y luego fue popularizado por Mike Cohn en su libro “Agile Estimating and Planning” [20]. Siendo esta una técnica relativamente nueva, existen pocos estudios empíricos [21].

Møløkken et. al. [5] realizaron un estudio donde aplicaron esta técnica de estimación en una empresa de software Noruega de tamaño medio que produce soluciones a medida para clientes del sector privado y público. Los autores comentaron que en algunos casos resulta ser más precisa que la combinación desestructurada de estimaciones en un grupo.

3 Un requerimiento de la gerencia: mejorar la estimación de PPS

La empresa donde se plantea la necesidad de mejorar la estimación de PPS es una multinacional con sede en Argentina. Al momento del estudio la empresa contaba con 400 empleados. La empresa no pertenece al rubro del desarrollo de software pero cuenta con un área de IT que da soporte a sus actividades comerciales. Dentro del área de IT, existe un área de desarrollo y soporte de aplicaciones que realiza desarrollo de software para la misma empresa.

La empresa ha definido dos diferentes procesos de desarrollo de software considerando principalmente el tiempo estimado, uno destinado a administrar Proyectos Grandes y otro para PPS. Los PPS son identificados principalmente por el valor del tiempo estimado, esto es el esfuerzo estimado en horas-hombre para llevarlo a cabo. Sintetizamos las características de los PPS en: a. Esfuerzo estimado mayor a tres días y menor a seis meses, b. Desarrollo de un sistema desde el requerimiento inicial, c. Desarrollo de nuevas funcionalidades, d. Mantenimiento de funcionalidades existentes; esto es, mejora de ciertas funcionalidades o corrección de errores, d. El grupo de desarrollo asume la responsabilidad desde el inicio de su desarrollo hasta su puesta en marcha.

Los proyectos grandes se estima que demandaran más de seis meses de esfuerzo, motivo por el cual se la asigna un Gerente de Proyectos que se hará cargo de dicho proyecto. Esto implica una demora significativa ya que activa otro proceso de administración y control de proyectos completamente diferente que demanda mucho más esfuerzo y por lo tanto tiene un mayor costo.

En general, los sistemas desarrollados en esta área son de tipo administrativos. Pueden ser desarrollados para plataformas web o de escritorio y cuentan con bases de datos relacionales e interfaces con otros sistemas de los cuales se toman los datos necesarios para el negocio.

En este contexto, la gerencia solicita que se revea el método de estimación usado habitualmente para los PPS. Es política de las empresas aceptar subestimaciones de PPS con un error menor al 10%. En caso de superar este porcentaje se debe re-evaluar el proyecto, con la finalidad de aprobar el aumento del presupuesto, calificarlo como un Proyecto Grande o cancelarlo.

La gerencia detectó un número excesivo de re-evaluaciones de PPS y una marcada insatisfacción en los clientes generada por incumplimientos en los productos. Una descripción detallada del proceso anterior a las modificaciones introducidas se puede consultar en [6].

Por lo tanto se plantearon los siguientes objetivos: a. Disminuir el número de re-evaluaciones , b. Mejorar la exactitud de las estimaciones (no superar el 10% del total estimado), c. Reconstruir y manejar la relación con clientes decepcionados.

4 Método de estimación basado en el juicio de expertos

Para mejorar el método de estimación de PPS se introducen una serie de modificaciones. En primer lugar se fortalece la relación con el cliente definiendo y consensuando con él los requerimientos. Se utiliza el estándar IEEE Std 830-1998 [23] por ser una guía para la especificación de requerimientos de software. Incluye la especificación de las características del sistema, requerimientos funcionales, prioridades, las interfaces externas -de usuario y externas- y los requerimientos no funcionales.

Una vez que se han consensuado los requerimientos las estimaciones se realizan sobre los requerimientos definidos. Se realizan rondas de estimación basadas en Juicio de Expertos, donde participan todos los miembros del equipo de desarrollo y se realizan ajustes de extremos, similares a los métodos Planning Poker, WideBand Delphi y Analogías.

Como parte del proceso de aprobación del presupuesto la gerencia analiza la contingencia del PPS y se aplica un factor de contingencia no mayor al 25%, aplicando los siguientes criterios: Contingencia del 15% (Complejidad baja, Desarrolladores familiarizados con el sistema, Criticidad baja), Contingencia del 20% **(**Complejidad media-alta, Desarrolladores familiarizados con el sistema, Criticidad media-alta), Contingencia del 25% (Complejidad alta, Algunos desarrolladores no están familiarizados con el sistema, Criticidad alta).

Al cabo de dos años de aplicar este método, se evalúa la eficiencia de su aplicación recopilando la información de ocho PPS. La Tabla1 muestra la descripción de los PPS analizados. También se detalla la cantidad de requerimientos para dar una idea del tamaño del PPS.

Tabla 1 – Descripción de los PPS

P_i	Descripción	Requerimientos
1	Cambios y corrección de errores funcionales	15
2	Cambios y corrección de errores funcionales	15
3	Cambios y corrección de errores funcionales y nuevos reportes	25
4	Nuevas funcionalidades críticas para el Cliente	25
5	Cambios y corrección de errores funcionales.	12
6	Nuevas funcionalidades y correcciones de errores	5
7	Se implementaron nuevas funcionalidades y un nuevo sistema de batch	25
8	Se implementó un nuevo módulo de control	26

La Tabla 2 describe los valores de la estimación. Incluyen las horas estimadas por proyecto (HE_i), que son iguales a la sumatoria de horas originales estimadas de cada requerimiento como se muestra en la fórmula (1):

$$HE_i = \Sigma \text{ HE Req1} + \text{HE Req2} + \ldots + \text{HE ReqN} \quad (1)$$

A este valor la gerencia suma un porcentaje de contingencia (Co%) basándose en las características de los proyectos y la experiencia del gerente. También se incluyen las horas reales de cada PPS (HR_i). Las HR_i son iguales a la sumatoria de horas reales (horas-personas) que insumió el desarrollo de cada requerimiento (ver fórmula 2)

$$HR_i = \Sigma \text{ HR Req1} + \text{HR Req2} + \ldots + \text{HR ReqN} \quad (2)$$

Además se muestra, el error relativo (ER) calculado en base a la fórmula 3. El signo de este valor muestra las desviaciones que se produjeron respecto de las estimaciones, pesimistas si son de signo negativo u optimistas si son de signo positivo.

$$ERco_i = [HR_i - (HE_i + Co\%)] / HR_i \quad (3)$$

También se muestra la magnitud del error relativo (MER) que es el valor absoluto del error relativo, según se describe en la fórmula 4.

$$MERco_i = ABS(ERco_i) \quad (4)$$

La última columna pone en evidencia los proyectos que necesitaron una re-evaluación. Aun considerando la contingencia como parte de las horas estimadas, vemos que los errores superan el 10% en la mayoría de los proyectos.

Tabla 2 - Estimación de proyectos

P_i	HR_i	$HE_i + Co_i\%$	$Co_i\%$	$ERco_i$	$MERco_i$	$ERco_i$ – 10%
1	130	116	20%	11%	11%	1%
2	159	128	20%	19%	19%	9%
3	667,5	560	20%	16%	16%	6%
4	478	415	0%	13%	13%	3%
5	99,5	120	25%	-21%	21%	-31%
6	76,5	66	25%	14%	14%	4%
7	498	435	25%	13%	13%	3%
8	539,5	490	25%	9%	9%	-1%
Medias			**20%**	**9%**	**14%**	**4%**

La Tabla 2 muestra que en 6 proyectos los errores superan el 10% lo que implica que no se ha evitado la re-evaluación de los mismos, las excepciones son el proyecto 5 y 8. El proyecto 5, que es el único proyecto que fue sobre-estimado, tiene la particularidad que no implementa una funcionalidad nueva sino que se focaliza en la corrección de errores. Esto podría justificar la actitud pesimista en las estimaciones, dado que cuando se estimó no se había analizado con profundidad las causas del error.

Según este planteo, para evitar una re-evaluación, sería necesario corregir la contingencia con la cual se trabaja. Se podría usar una contingencia de un 34%, siendo la contingencia promedio usada anteriormente de un 20% a lo que podríamos sumarle una corrección de un 14% que es la media de la magnitud del error relativo.

En la Tabla 3 presentamos un escenario más claro donde quitamos la contingencia de las horas estimadas ya que queremos medir puramente la exactitud en las estimaciones de esfuerzo. Vemos que los valores de MRE están en un rango de [10%-36%]. Aplicamos porcentajes de Co en un rango de 34% al 40% (cfr. Tabla 4), observamos que un valor del 39%, nos asegura que todos los MRE no superan el 10%, asegurando que no van a requerir los PPS re-evaluaciones.

Cabe destacar que una corrección del 39% de Co aumenta los valores de sobreestimación, aspecto que también tiene sus consecuencias negativas al provocar horas ociosas o una disminución de la productividad del grupo de trabajo.

Tabla 3 - Estimación de proyectos SIN considerar la contingencia (Ca%)

P_i	HRi	HEi	ERi	MREi
1	130	92,8	29%	29%
2	159	102,4	36%	36%
3	667,5	448	33%	33%
4	478	415	13%	13%
5	99,5	90	-10%	10%
6	76,5	49,5	35%	35%
7	498	326,25	34%	34%
8	539,5	367,5	32%	32%

Tabla 4. MER calculados usando diferentes valores de Ca%

P_i	34%	35%	36%	37%	38%	39%	40%
1	4%	4%	3%	2%	1%	1%	0%
2	14%	13%	12%	12%	11%	10%	10%
3	10%	9%	9%	8%	7%	7%	6%
4	-16%	-17%	-18%	-19%	-20%	-21%	-22%
5	-21%	-22%	-23%	-24%	-25%	-26%	-27%
6	13%	13%	12%	11%	11%	10%	9%
7	12%	12%	11%	10%	10%	9%	8%
8	9%	8%	7%	7%	6%	5%	5%

A modo de conclusión destacamos:

a) No se evitaron los cambios de alcance ya que todos tuvieron errores mayores al 10%.
b) El cliente está más satisfecho debido a que se cumplió con la entrega de los requerimientos acordados para cada proyecto.
c) Para la re-evaluación de los PPS sería necesario aplicar una contingencia de 39% para quedar dentro de los límites aceptables menores al 10%.

5 Discusión sobre el método aplicado

En el presente caso de estudio planteamos un nuevo método de estimación sobre requerimientos completos y consensuados con los clientes. Con respecto al método anterior se han introducido las siguientes modificaciones:

a) Se deja de usar una tabla previamente definida donde se establece el valor de las horas estimadas de acuerdo con la duración prevista (en días) y su complemento que es una plantilla de estimaciones, que es exigida en la aprobación de una re-evaluación. Ambas son documentos de referencia de

acceso público a los desarrolladores. Se simplifica el proceso. Para más detalle ver [6].

b) Se realiza una especificación de requerimientos más profunda, facilitada por el uso del IEEE Std 830-1998 [23]. El uso de esta herramienta facilito la apertura de los canales de comunicación con el cliente y la documentación de los requerimientos acordados y sus posibles soluciones como parte del contrato, otorgando transparencia al proceso. Las estimaciones se realizan sobre una definición más clara de los requerimientos.
c) Se estima aplicando juicio de expertos convocando a todas las personas responsables del desarrollo.
d) Se ajusta el valor de contingencia.

Los proyectos involucrados en la muestra son ejemplos de proyectos tipos del área, de los cuales se disponían los datos. Abarcan proyectos muy pequeños de unas 100 horas reales de esfuerzo hasta proyectos de más de 650 horas-persona de esfuerzo.

Desde el punto de vista operativo, no se lograron estimaciones lo suficientemente exactas como para no exceder el límite de tolerancia del 10%. Pero sí se logró cumplir con las expectativas de los clientes y evitar conflicto con los mismos al establecer un acuerdo respecto de los requerimientos completos y las posibles soluciones que se iban a dar a cada uno de ellos.

Vimos que sería bueno hacer una corrección de la contingencia a aplicar sobre las estimaciones de esfuerzo de un 39%, dejando al líder de proyectos la evaluación del riesgo de sobreestimación, el cual es posible que ocurra.

No sería recomendable dejar las estimaciones libradas sin contingencia ya que vimos que en la mayoría de los casos, los estimadores fueron optimistas y subestimaron las tareas. Como contrapartida, tenemos que la sobreestimación puede generar estimadores pesimistas que hagan valer la Ley de Parkinson que sostiene que mientras más tiempo se tiene, más tiempo toma una tarea en realizarse.

Según Jørgensen et. al. [24] uno de los efectos negativos que más influyen en el fracaso de los proyectos de software es el exceso de optimismo. Descubren un punto importante que pueden estar llevando a los estimadores a ser demasiado optimistas, esto es el formato utilizado al formular la pregunta que solicita la estimación de esfuerzo. El formato tradicional sería: "¿Cuántas horas se necesitan para completar la tarea X?" y el alternativo sería: "¿Cuántas tareas se pueden completar en Y horas?" Al analizar la situación que se daba en la empresa antes introducir las modificaciones vimos claramente que se optaba por el formato alternativo, donde los clientes planteaban al equipo de desarrollo cuántas tareas (o requerimientos) se podían cumplimentar dado un presupuesto anual previamente determinado. La recomendación de los autores es que siempre se opte por el formato tradicional ya que este no conlleva ninguna desviación impuesta por los clientes que quieren obtener el máximo con un presupuesto irreal. El escenario descripto logró revertirlo, sacando el foco del presupuesto disponible y dirigiéndolo al trabajo conjunto con el cliente a través de la toma de requerimientos, estimaciones y priorización de tareas. El conocimiento del presupuesto disponible solo estableció el límite sobre la ejecución de las tareas rankeadas.

Reconocemos que quizás el margen de error aplicado podría ser mayor si contáramos con otro equipo de trabajo [26], dado que el equipo varió muy poco de proyecto a proyecto y los desarrolladores eran personas calificadas como semi-senior quienes además conocían los sistemas.

Recomendamos trabajar con los clientes en el entendimiento de cada requerimiento, planteo de posibles soluciones y estimar en grupo antes de la aprobación presupuestaria. Aspectos que coinciden con los métodos ágiles, si bien no ha sido adoptada la metodología ágil para el desarrollo de los PPS. Al mismo tiempo, estas definiciones deben quedar plasmadas en el contrato y en el documento de requerimientos para evitar conflictos y cubrir las expectativas. Esto otorga transparencia al proceso.

También destacamos la importancia que la toma de requerimientos, para la ejecución de PPS de software, sea realizada por el líder del proyecto para evitar que las estimaciones de esfuerzo se vean altamente afectadas por información irrelevante y desconcertante. Jørgensen y Grimstad [25] aconsejan evitar estos efectos eliminando o neutralizando esta información irrelevante en la especificación de requerimientos antes de ser conocida por los estimadores. Es difícil volver al estado mental anterior luego de haber sido expuesto a este tipo de información.

También nos planteamos que si se hubiera aumentado la contingencia en el método de estimación anterior a los cambios, hubiera sido posible resolver el problema planteado. Lamentablemente al no tener datos históricos sobre la aplicación del método no es posible afirmarlo ni negarlo. Al mismo tiempo destacamos que en el hipotético caso de que se pudiera disminuir la frecuencia de revisiones de PPS creemos que hubiera sido poco probable que se mejore la relación con el cliente, que era otro aspecto solicitado.

6 Conclusión

Para poder contestar la pregunta de investigación de si es posible mejorar la estimación de esfuerzo medido en horas persona de los PPS, investigamos los diferentes métodos de estimación de productos de software basados en el juicio de expertos. Estos conocimientos nos ayudaron a resolver el problema que se presentó en una empresa multinacional cuando se planteó la necesidad de disminuir el porcentaje de error de PPS para evitar las revisiones sucesivas de presupuesto. Para esto diseñamos un método que se aplicó a un conjunto de ocho proyectos típicos reales de la empresa, durante aproximadamente dos años.

La aplicación de este nuevo método de estimación fue positiva, aunque para proyectos futuros resta ajustar el valor de la contingencia a aplicar para lograr la reducción del error según el estándar definido por la empresa. Hicimos una prueba aplicando una progresión de contingencias sobre los valores de horas persona estimadas hasta obtener errores menores al límite permitido en la empresa. Esto nos dio como resultado que siempre debemos considerar una contingencia del 39%.

El ejercicio de tomar datos históricos y analizar las diferencias ha sido un entrenamiento necesario para resolver el problema planteado y lograr una mayor eficiencia en la estimación futura de proyectos de software.

A modo de resumen:

a) Sugerimos la incorporación de una contingencia de 39%. También recomendamos que el líder de proyectos realice una evaluación del riesgo de sobreestimación; dado que es posible que esto ocurra.
b) La diferencia a destacar entre la aplicación del método propuesto con respecto al anterior es la estimación basada en expertos y la satisfacción del cliente, ya que el contrato con el cliente incluye tanto un presupuesto otorgado como la entrega de funcionalidades acordadas sobre la base de contar con requerimientos completos. Ambos conceptos no se pueden desligar.

Como conclusión final se estableció que es posible evitar las revisiones presupuestarias aumentando la contingencia y se pudo establecer un entorno de trabajo en equipo saludable y de amplia colaboración con el cliente, de allí que recomendamos su implementación.

Es necesario investigar la aplicabilidad (o modificaciones a realizar) del método planteado en proyectos grandes con mayor número de desarrolladores, el impacto en las estimaciones al aplicarlo sobre proyectos trabajados por equipos heterogéneos. Es decir, equipos constituidos por desarrolladores con distintos niveles de experiencia y por desarrolladores que tengan distintas experiencias previas en el desarrollo del sistema en cuestión.

Agradecimientos. La realización de este trabajo ha sido posible gracias a la financiación de la Universidad Austral.

Referencias

1. Charette, R.: Why software fails? IEEE Spectrum (September): (2005) 42-49
2. Jørgensen, M.: A review of studies on expert estimation of software development effort. Journal on System and Software, Vol. 70, No. 1-2, (2004) 37-60
3. Jørgensen, M. and Shepperd, M. : A systematic review of software development cost estimation studies. IEEE Transactions on Software Engineering, Vol. 33, No. 1, 3-53, Janu-ary (2007)
4. Rowe, G and Wright,: Expert Opinions in Forecasting: The role of the Delphi Technique. In Principles of forecasting Springer, US (2002) 125-144
5. Moløkken-Ostvøld, K., Haugen N.C., Benestad H.C.: Using planning poker for combining expert estimates in software projects. Journal of Systems and Software, 81 (12), (2008) 2106-2117
6. Santos, Silvana. Estimación de proyectos de software pequeños basada en el juicio de expertos. Tesis de Maestría UNLP, (2014) DOI: http://hdl.handle.net/10915/41260

7. Hihn, J., Habib-Agahi, H.: Cost estimation of software intensive projects: A survey of current practices. International Conference on Software Engineering, IEEE Comput. Soc. Press, Los Alamitos, CA, USA (1991) 276-287
8. Heemstra, F. J.,. Kusters, R. J: Function point analysis: Evaluation of a software cost estimation model. European Journal of Information Systems 1(4): (1991) 223-237
9. Kitchenham, B., Pfleeger, S. L., McColl, B.. Eagan, S.: An empirical study of maintenance and development estimation accuracy. Journal of systems and software, 64(1), (2002) 57-77
10. Boehm, B.: Software Engineering Economics. Prentice Hall (1981)
11. Jørgensen, M.: An empirical evaluation of the MkII FPA estimation model. Norwegian Informatics Conference, Voss, Norway, Tapir, Oslo (1997) 7-18
12. Boehm, B., C. Abts, and Chulani, S.: Software development cost estimation Approaches - A survey. Annals of Software Engineering, 10: (2000) 177-205
13. Wiegers , Karl, E.: Stop Promising Miracles. Software Development magazine. (2000)
14. Shepperd, M., Schofield, C.: Estimating Software Project Effort Using Analogies. IEEE Transactions on Software Engineering, Vol. 23, N° 12, November (1997)
15. Vicinanza, S., Mukhopadhyay, T., Prietula, M.: Examining the feasibility of a case-based reasoning model for software effort estimation, MIS Quarterly, 16 (June), (1992) 155-71
16. Vicinanza, S., Mukhopadhyay, T., Prietula, M.: Software Effort Estimation With a Case-Based Reasoner. J. Experimental & Theoretical Artificial Intelligence, 8, (1996) 341 – 363
17. Vicinanza, S., Mukhopadhyay, T., Prietula, M.: Case-based reasoning in effort estimation. In Proc. 11th Intl. Conf. on Info. Syst. (1990)
18. Kadoda, G., Cartwright, M., Chen, L. and Shepperd, M.: Experiences using Case-Based Reasoning to predict software project effort. In Proceedings of the EASE conference keele, UK, (2000)
19. Atkinson, K., & Shepperd, M. J. (1994). The use of function points to find cost analogies. DOI: http://bura.brunel.ac.uk/handle/2438/1553
20. Cohn, M.: Agile Estimating and Planning. Robert C. Martin Series. Prentice Hall (2005)
21. Haugen, N.C.: An empirical study of using Planning Poker for User Story estimation. In Proceedings of AGILE Conference. Computer Society. IEEE (2006)
22. Grenning, J.: Planning Poker or How to avoid analysis paralysis while release planning. Hawthorn Woods: Renaissance Software Consulting, vol. 3. (2002) DOI: http://sewiki.iai.uni-bonn.de/_media/teaching/labs/xp/2005a/doc.planningpoker-v1.pdf
23. IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications, (1998) DOI: https://standards.ieee.org/findstds/standard/830-1998.html
24. Jørgensen, M. and Halkjelsvik, T.: The effects of request formats on judgment-based effort estimation, Journal of Systems and Software, 83 (1), (2010) 29-36.
25. Jørgensen, M. and Grimstad, S.: The Impact of Irrelevant and Misleading Information on Software Development Effort Estimates: A Randomized Controlled Field Experiment. IEEE Transactions on Software Engineering,. IEEE computer Society Digital Library. IEEE. (2010)
26. Ktata, O and Lévesque, G.: Designing and implementing a Measurement Program for Scrum Teams: what do agile developers really need and want? In Proceedings of the Third C* Conference on Computer Science and Software Engineering ACM (2010) 101-107