Los Escenarios Principales como Elementos Básicos de Estimación

José Ignacio Cao, Eduardo Diez, Paola Britos y Ramón García-Martínez
Centro de Ingeniería de Software e Ingeniería del Conocimiento. Escuela de Postgrado. ITBA.
Argentina
jignaciocao@hotmail.com, {ediez. pbritos, rgm}@itba.edu.ar

Abstract

The early and accurate estimation of software projects is a hard, almost impossible, task. Nevertheless most of the third party developed systems require fixing a price before construction in order to settle the economical agreement. Then, an early estimation is necessary to define that price.

This paper proposes the estimation of use cases comparing their scenarios with standard scenarios. The paper also explains the disadvantage of other popular estimation methods.

Keywords: software estimation , size , effort , use cases, scenarios.

Resumen

La correcta estimación temprana de un proyecto de software es una tarea difícil o casi imposible. Sin embargo la mayor parte de los sistemas que son desarrollados por terceros requieren fijar un precio antes de contratarse el desarrollo. La estimación es necesaria para la definición de ese precio.
El presente trabajo propone la estimación de casos de uso por comparación de sus escenarios con escenarios normalizados, explicando, además, la desventaja de otros métodos utilizados.

Palabras claves: estimación, tamaño, esfuerzo, casos de uso, escenarios.

1 INTRODUCCIÓN

1.1. El problema de la estimación temprana

En el marco de la gestión de proyectos puede definirse la estimación como “el proceso que proporciona un valor a un conjunto de variables para la realización de un trabajo, dentro de un rango aceptable de tolerancia.” [1].
Los problemas de la estimación son varios. Sabemos que estimar significa, de alguna forma, predecir el futuro, actividad con una incertidumbre que se refleja en la frase “rango aceptable de tolerancia” de la definición dada. Una de las formas de predecir el futuro es tomar en cuenta lo que sucedió en el pasado. En ese sentido, las técnicas de estimación se basan de una forma u otra en datos históricos y experiencias previas sobre elementos o subelementos similares al que se debe estimar.
Surge de esto un primer problema, la identificación de elementos o subelementos similares o comparables a lo que debemos estimar. Ahora bien, la actividad de estimación no se hace una sola vez en el proyecto. A medida que se cuenta con más datos se hacen estimaciones más precisas que nos permiten una mejor planificación de lo que resta del proyecto. De todos los puntos en los cuales puede realizarse la estimación, cuando menos datos tenemos para hacerla es en el momento inicial, cuando todavía se está evaluando la factibilidad económica del proyecto. Desde el punto de vista del desarrollo y venta de software específico para terceros, esa estimación, que llamaremos temprana, es una de las más importantes porque fija el precio de contratación.
Si nuestro objetivo es medir el software debemos establecer algún elemento o parámetro básico de estimación a usar en una métrica. Es decir, un elemento básico para representar el software, o parte de él, el cual deberá ser susceptible de asignársele algún valor para llegar a un número final que expresará en forma cuantitativa la medida de ese software.
El objetivo de este trabajo es demostrar la conveniencia del uso de escenarios normalizados como elemento básico de comparación con escenarios reales, tanto para la estimación temprana como en cualquier otro punto del proceso. Por otra parte, para aclarar algún punto o enfatizar su factibilidad, nombraremos la experiencia de la empresa argentina Idea Factory Software (empresa evaluada CMM nivel 3), quién desarrolló un método de estimación basado en estos principios y que utiliza con éxito desde hace un año y medio.

2. INCONVENIENTES DE LA TÉCNICA DE PUNTOS DE FUNCIÓN

Una medida del tamaño de un sistema, de uso generalizado son los puntos de función. El concepto de punto de función fue presentado en 1979 (primera publicación) por Allan Albrecht refinándolo en 1984. El método intenta medir el tamaño de un sistema software.
El presente trabajo no tendría sentido si pensáramos que los puntos de función solucionan el problema planteado. A los efectos de una estimación temprana los puntos de función tienen el inconveniente de asumir que se conocen los grupos de datos que utilizará el sistema y además asume que ese conocimiento es lo suficientemente amplio como para calificar esos grupos en complejidades baja, media y alta.
En general en el período de preventa no se dispone de esos datos, especialmente de los datos internos del sistema. Ese es el primer problema, casi insuperable. Por otra parte, si se dispusiera de esa información es complicado en tiempo y costo hacer el análisis necesario para calcular los puntos de función.
Podría quizás utilizarse un método simplificado evaluando sólo las entradas, salidas y consultas. En ese caso utilizaríamos la información funcional disponible, aunque el modelo quedaría demasiado simple para representar la realidad.

Por otra parte hay una consideración adicional. En la actualidad el análisis, diseño y desarrollo de sistemas se hace principalmente con metodologías orientadas a objetos. La técnica de los Casos de Uso, si bien no es privativa de la orientación a objetos, es claramente preponderante. El desarrollo del sistema es conducido por casos de uso [2] y los casos de uso se utilizan como unidades tanto de análisis, diseño, pruebas como de desarrollo.
Si utilizáramos puntos de función deberíamos hacer un análisis (entradas, salidas, consultas, datos) que sólo se usa para esta técnica. Sería más eficiente hacer un análisis que además de servir para la estimación coincida con la metodología utilizada. Esto es importante si se tiene en cuenta que deberá reestimarse varias veces en el transcurso del proyecto para obtener las diferentes métricas de gestión, con lo cual no es eficiente hacer cada vez una conversión de una unidad de trabajo en otra (puntos de función a casos de uso). En pocas palabras, si nuestra unidad de trabajo es el caso de uso, es más natural contabilizar casos de uso en lugar de entradas, salidas, consultas y datos.

3. LOS CASOS DE USO COMO ELEMENTO BÁSICO INADECUADO

3.1. Casos de uso como elemento básico – Método Use Case Points

Como ya dijimos en la sección anterior, dado que los casos de uso son la unidad rectora en la construcción de sistemas orientados a objetos, es natural considerar el caso de uso como un elemento o parámetro básico apto para el proceso de estimación.
Un ejemplo de este criterio es el difundido método Use Case Points desarrollado sobre el trabajo de Gustav Karner en 1993 [3] y que recrea de alguna forma el trabajo de Allen Albrecht sobre puntos de función. El método Use Case Points clasifica los casos de uso en simple, promedio y complejo con factores de peso 5, 10 y 15 respectivamente. La clasificación se hace en base al número de transacciones que contiene el caso de uso, 1 a 3 para simple, 4 a 7 para promedio y 8 ó más para complejos. Por otra parte se define una transacción como un evento que ocurre entre un actor y el sistema a ser modelado. A nuestro juicio, esta clasificación es totalmente inadecuada a los efectos de realizar una estimación.
El principal motivo de esto es que un caso de uso no tiene un tamaño determinado. En un ensayo para Rational Software, John Smith [4] considera como normal un promedio de 30 escenarios por caso de uso, tratando de ser conservador en el número. Un caso de uso de este tipo podría llegar a tener 30 transacciones. Si a cada escenario le correspondiera una transacción, este caso de uso tendría igual factor de peso que uno de 8 transacciones. Es decir el rango superior, 8 a infinito, da igual valor para 8 transacciones que para infinitas, lo cual es un inconveniente.
Ahora bien, los inconvenientes persisten si se mejora la escala asignando puntos de caso de uso proporcionalmente al número de transacciones. En efecto la complejidad de un escenario puede ser tanto como la complejidad de otros 10, ó 20 ó el número que se quiera elegir y lo mismo vale para las transacciones. Por otra parte la agrupación de los requerimientos en casos de uso puede ser totalmente arbitraria haciendo que el método dé valores muy distintos para el mismo sistema. En ese sentido hemos visto un trabajo sobre el método de Use Case Points donde para hacer un programa de altas, bajas, modificaciones y consultas resultó un esfuerzo de programación de 4 meses utilizando el lenguaje C++. Para un programa de altas, bajas, modificaciones y consultas, cuatro meses es a todas luces excesivo. En ese caso el método fue bien aplicado, sin embargo se consideró un caso de uso para el alta, otro para la baja, otro para la modificación y otro para la consulta.
Hoy en día podemos decir, como explica Bittner [5] que semejante división es un error, que alta, baja, modificación y consulta no conforman cuatro casos de uso. Sin embargo en el libro Writing Effective Use Cases [6], quizás ya fuera de época, podemos leer que la opción de hacer un caso de

uso para cada una de esas funciones, alta, baja, modificación y consulta, es una alternativa válida. Por otra parte hay quien, con criterio razonable, establece un caso de uso con el alta, y otro con la consulta, siendo baja y modificación extensiones de esta. Es decir, en algo tan simple y común como un programa de mantenimiento podemos considerar uno, dos o cuatro casos de uso, haciendo que nuestra estimación llegue casi hasta cuadruplicarse según la separación que hagamos.
En conclusión, un método basado en casos de uso debería entonces tener en cuenta el número de transacciones o escenarios del mismo para determinar su peso. Sin embargo teniendo en cuenta esto no resolveríamos el problema de la complejidad que pueda tener el conjunto de escenarios contenidos en cada caso de uso, es decir no es sólo un problema de cantidad sino también de complejidad.

3.2. Las Clases del Sistema como alternativa a los Casos de Uso

Descartando los casos de uso, otra unidad compatible con la orientación a objetos podría ser el de las clases involucradas en el sistema. La metodología española Métrica 3 [7] en su sección "Técnicas y prácticas" sabiamente elude el caso de uso como parámetro de estimación recomendado y plantea la posibilidad de utilizar el método Staffing Size. Define este método como "un conjunto de métricas para estimar el número de personas necesarias en un desarrollo Orientación a Objetos, y para determinar el tiempo de su participación en el mismo. Las unidades que utiliza son el número de clases clave y clases secundarias existentes en el modelo y toma en cuenta el lenguaje de programación utilizado. En este caso, al igual que los casos de uso, las clases clave pueden llevar asociadas funciones más o menos complejas y con más o menos operaciones a programar.
En conclusión algunos de los inconvenientes indicados para los casos de uso subsisten si se utilizan las clases como parámetro de comparación.

3.3. Los Casos de Uso - Unidad de Agrupación

Como dijimos, los casos de uso son la unidad de trabajo en los métodos orientados a objetos que los utilizan. En este sentido es importante que la estimación también pueda expresarse en casos de uso. Si bien afirmamos en esta sección que los casos de uso no deben ser el parámetro básico para la estimación, eso no quita, que no siendo el parámetro básico, sí puedan ser una unidad de agrupación.

4. LA FALACIA DEL TAMAÑO INDEPENDIENTE DE LA TECNOLOGÍA

4.1 El Tamaño y el Esfuerzo

Hasta este momento hemos hablado de estimación, pero, en lo posible, evitamos cuidadosamente aclarar si nos referimos a estimación de tamaño o de esfuerzo. No es posible seguir adelante sin hacer algunos comentarios al respecto, principalmente porque al hablar de estimación cualquier interlocutor educado en la teoría, pregunta si es estimación de tamaño o esfuerzo, y no acepta seguir hablando si no se hace esa aclaración.
Esta actitud cuasi dogmática, que se observa a menudo en los mejores profesionales, justamente por ser los mejores, se debe a que la posibilidad de cuantificar puramente el tamaño de un sistema, es un dogma que, como tal, sólo depende de la fe del que lo esgrime.

4.2. Un Sistema no es una mesa

En efecto, si hablamos de construir una mesa tenemos claro qué es el tamaño. Puede ser una mesa de 1m de altura, con una tabla de 1m por 2m. Podemos dar el volumen en m3 de la madera a utilizar o dar su peso esperado como medida del tamaño. Por otra parte el esfuerzo en construirla se medirá en horas. Evidentemente el esfuerzo será dependiente de la tecnología a utilizar. No es lo mismo hacer una mesa con sierra manual que con sierra eléctrica. Por otra parte la habilidad del carpintero y sus condiciones de trabajo también influirán en la cantidad de horas que insumirá la construcción. Entonces el tamaño es un atributo de la mesa y el esfuerzo depende del tamaño pero también de otros factores (tecnológicos, de habilidad, entre otros).
Este esquema simple se intenta utilizar en el desarrollo de sistemas. Se mide el tamaño, el cual se considera independiente de la tecnología como primer atributo del mismo, y en función de ese tamaño se calcula el esfuerzo para una determinada tecnología y condiciones de trabajo.
El inconveniente comienza cuando intentamos medir el sistema para determinar su tamaño.
Evidentemente ni los centímetros ni los kilogramos utilizados para la mesa son útiles para medir el tamaño de un sistema, especialmente si aún no fue construido. Es necesario entonces identificar una unidad de medida del tamaño del sistema y aquí se pierde la independencia de la tecnología y se desdibuja el puro concepto de tamaño. Las unidades que inicialmente se nos puedan ocurrir terminan siendo dependientes de la tecnología utilizada, líneas de código, tamaño en bytes etc.
Podemos decir que el tamaño de un sistema está relacionado con la funcionalidad del mismo. En ese sentido nada parece más indicado que utilizar algo así como “puntos de función” para medir la funcionalidad. El más popular exponente de este sistema es el método de puntos de función de Albretch. De este método se afirma que mide tamaño independiente de la tecnología. El método dice, por ejemplo, que una entrada de complejidad media tiene 4 puntos de función y que a una consulta media también le corresponden 4 puntos de función. Ante esta afirmación surge una pregunta, ¿por qué la función de entrada de datos es equivalente funcionalmente a la de consulta? ¿qué criterio se utilizó para establecer esa equivalencia entre estas dos funciones y todas las demás que conforman el método?
Albretch estudió 24 proyectos de aplicaciones de negocios con un rango de tamaño desde 3000 a 318.000 líneas de código desarrolladas en DMS, PL/1 y COBOL.
Podemos asumir, dado que no encontramos una explicación, que en esos proyectos Albretch halló una relación entre entrada y consulta que le permite decir que la cantidad de líneas de código utilizadas para una entrada de complejidad media es igual a la cantidad de líneas de código de una consulta media.
Sin embargo tenemos que reconocer que esa igualdad de líneas de código se da para los lenguajes analizados y no necesariamente para cualquiera.
En efecto, si un lenguaje tiene facilidades para programar una consulta entonces esa relación ya no existe. Vemos en consecuencia que el tamaño, “independiente de la tecnología” del método de puntos de función ya no es tan independiente de la tecnología, y por lo tanto no sería "tamaño" según la definición clásica, sino "esfuerzo normalizado" para una o más tecnologías.

4.3. Tamaño no fácilmente cuantificable

Todo esto se puede generalizar a cualquier tipo de métrica que pretenda medir tamaño funcional reduciendo a una unidad común los diferentes tipos de funciones. La única forma de definir pura y estrictamente tamaño, independientemente de cualquier tecnología, es enumerar la cantidad de funciones de un mismo tipo que tiene un sistema, por ejemplo decir que el tamaño es 20 altas simples, 30 modificaciones complicadas, 12 listados simples , etc. , pero al tratar de establecer una

equivalencia entre un tipo función y otra, necesariamente se acude a nociones relacionadas con el esfuerzo de construcción de las mismas (codificación, diseño,análisis, etc.), ya que las únicas unidades de medida que tiene en común un alta con una modificación o un listado son las horas que se tarda en construirlas o también el tamaño físico (sean bytes o líneas de código o páginas de documentación) que tienen en una determinada tecnología o método.
En conclusión, en el método que proponemos, cuando decidimos cuantificar escenarios estaremos hablando de "tamaño, en cierta medida, dependiente de la tecnología", que también podemos llamar "esfuerzo normalizado", es decir definido y medido definiendo como fijos y standard un conjunto determinado de factores tecnológicos. Como ejemplo de esto, en la implementación práctica de este método realizada por Idea Factory Software, fue necesario realizar una distinta escala de valores para una tecnología (Oracle Forms) en donde las relaciones de esfuerzo entre las funciones variaba notablemente con respecto a las otras (J2EE y .Net).

5. ESCENARIOS PRINCIPALES COMO ELEMENTO BÁSICO ADECUADO

5.1. Tres Problemas: Normalización, Cuantificación y Extrapolación

En la sección 3 de este trabajo hemos postulado que los casos de uso no son un elemento básico adecuado para la estimación. Por otro lado son la unidad de trabajo natural en un desarrollo de software orientado a objetos. Debemos encontrar un elemento básico de estimación que salve los inconvenientes descriptos anteriormente, que sea agrupable por caso de uso, para poder controlar la evolución de cada caso de uso, y a la vez que implique un esfuerzo de análisis que luego pueda ser aprovechado.
El tamaño de un sistema depende de su funcionalidad. Funcionalidad es un término que de alguna manera significa lo que el sistema puede hacer, en suma, el conjunto de sus funciones. Como expusimos en el capítulo anterior, cuantificar el tamaño implica algunos inconvenientes y asunciones que deben hacerse. En ese esquema hay una solución para llegar a la estimación si se resuelven tres problemas principales:

a) ¿Cómo clasificar en forma normalizada las funciones ("funcionalidad") de un sistema? **(normalización)**
b) ¿Qué valores numéricos comparativos se deben asignar a cada función normalizada? **(cuantificación)**
c) ¿Qué relación hay entre una unidad de medida de cada función (item b) y la unidad de esfuerzo de la actividad que se quiere estimar **(extrapolación)**

5.2. Los Escenarios como elementos básicos de estimación

Ahora bien, si queremos tener un esquema de estimación compatible con nuestra metodología orientada a objetos, deberíamos buscar en esa metodología qué elemento será la unidad básica de estimación en nuestro sistema.
En orientación a objetos las funciones se traducen en operaciones contenidas por las clases. Pero no se llegan a identificar las operaciones hasta bien avanzado el proyecto, por lo cual no podrían usarse como elementos de estimación en una etapa temprana. Por otro lado su número y granularidad es tal que se harían casi inmanejables.
El nivel superior a la operación, en cuanto a elementos funcionales de la orientación a objetos, es el escenario, y el nivel superior a ese es el caso de uso. Hemos hecho consideraciones por las cuales

no consideramos apropiada la utilización de casos de uso, por lo cual deberemos analizar el uso de escenarios. Formalmente un escenario representa una forma de uso del sistema, no es exactamente una función, pero sí representa la funcionalidad del sistema. Por otra parte, si sabemos lo que el sistema debe hacer, entonces podemos saber cuáles serían sus escenarios principales. Esta última afirmación puede ponerse en duda a priori, sin embargo la experiencia práctica realizada por Idea Factory Software tuvo repetidamente éxito en demostrar este punto.
Ahora bien, es evidente que la cantidad de escenarios principales podría servirnos para el cálculo del tamaño/esfuerzo relacionado con un desarrollo, pero debería considerarse la complejidad del escenario para que esa medida fuera proporcional a la funcionalidad.
Es necesario aquí detenerse para aclarar algo sobre los escenarios principales, dado que la teoría y la práctica son un tanto ambiguas al respecto. Por un lado se puede considerar que un caso de uso tiene uno y sólo un escenario principal y luego escenarios alternativos. Es algo que inicialmente tiene sentido y parte de la teoría coincide con esta idea. Por otro lado, si como dijimos en la sección 3, se tiende a que las funciones de mantenimiento de una clase, alta – baja – modificación y consulta, deban agruparse en un sólo caso de uso ¿podemos decir que hay un escenario principal , alta por ejemplo, y que consulta es una alternativa?. Esto sería raro dado que el valor agregado obtenido por el usuario difiere en cada escenario y ambos escenarios no tienen casi ningún paso en común.
Es por eso que consideramos la postura que contempla que un caso de uso puede tener más de un escenario principal, cada uno con sus eventuales escenarios alternativos. Esta postura está sustentada en la teoría más pura si consideramos que en el libro básico de UML [8] dice que “Para cada caso de uso, usted encontrará escenarios primarios (los cuales definen secuencias esenciales) y escenarios secundarios (los cuales definen secuencias alternativas)” , declaración esta que contempla la existencia de más de un escenario principal, o primario, por caso de uso.
En función de todo lo expuesto vemos que los escenarios principales cumplen las características necesarias para ser elementos básicos para la estimación de un sistema. En efecto, no dependen de cómo se agrupan los escenarios en los casos de uso, pero permiten expresar la estimación en función de estos. Representan la funcionalidad del sistema, con información disponible en etapas tempranas del desarrollo y además la clasificación en escenarios no es trabajo adicional ya que de todos modos debe hacerse para un análisis orientado a objeto. Son además un elemento fácilmente identificable, apto para ser valorado numéricamente y para ser refinado a medida que se obtiene mayor información.

5.3. Normalización

Corresponde ahora resolver el tema de la clasificación normalizada de esos escenarios. Para esto será necesario poder clasificar los escenarios principales en una forma standard y repetible.
Una de las soluciones es entonces encontrar una lista de escenarios típicos, conocidos por todos los estimadores, y que de alguna forma puedan asimilarse a (o más bien compararse con) cualquier escenario que encontremos en la realidad. Además, para una mejor clasificación relacionada con la complejidad, cada escenario de esa lista podrá tener subtipos definidos en función de distintos atributos de los mismos. Por ejemplo puede existir un escenario de tipo alta con subtipo simple (en función del número aproximado de campos involucrados, por ejemplo) , subtipo normal y subtipo cabecera - detalle. Para cada tipo - subtipo deberá especificarse el flujo normal y cuáles alternativos y excepciones se consideran standard, a fin de poder compararlo con el escenario que corresponda de la vida real.
Ahora bien, ¿es posible hacer una lista de escenarios normalizados de modo tal que cualquier escenario de un sistema real pueda asimilarse a un escenario de la lista?. Basados en el método utilizado por Idea Factory Software podemos decir que la respuesta es sí. En la implementación

realizada por esta empresa se utiliza una lista de 11 escenarios normalizados que cubren todos los escenarios posibles, al menos en sistemas de gestión administrativa, contable y financiera en donde el método fue aplicado.

5.4. Desnormalizando la normalización - Complejidades y Facilidades adicionales

Por otra parte, es evidente que en el mundo real los escenarios difieren en mayor o menor medida de los escenarios normalizados. Si no se quiere tener una lista extensa de escenarios normalizados podemos buscar un elemento de ajuste para esos escenarios. Por ejemplo, nosotros podemos decir que un determinado escenario se parece bastante a la definición de un escenario que llamamos “Alta” con subtipo “simple”, pero sin embargo, desde el principio identificamos que ese escenario principal tiene una gran cantidad de escenarios alternativos. Entonces se toma en cuenta una “complejidad” que represente esa característica y que sume puntos de escenario, quizás en función de la cantidad aproximada de escenarios alternativos que se estiman para el escenario real.
Por otra parte puede existir una condición que haga que un escenario de la vida real sea más fácil que uno normalizado (por reuso de componentes, por ejemplo). En ese caso se involucraría una “facilidad” que reste puntos de escenario. En pocas palabras las complejidades/facilidades cubren la diferencia que pueda existir entre el escenario de la vida real y el, o los, escenarios normalizados utilizados para representarlo.
Sabemos que la estimación es necesariamente un proceso que puede tener grandes variaciones y entonces es posible que los ajustes a los escenarios normalizados por complejidades o facilidades se consideren irrelevantes. En todo caso sólo deben usarse para cubrir diferencias significativas entre el escenario real y el normalizado.
Si se quieren usar complejidades y facilidades, es evidente que, además de la lista de escenarios normalizados, deberá identificarse una lista de complejidades o facilidades más o menos comunes y cuantificarlas. Si bien es fácil hacer una lista corta de escenarios base , no es tan fácil cubrir todo el espectro de las complejidades y facilidades, por lo cual debería dejarse a criterio del estimador crear y cuantificar alguna complejidad/facilidad que pueda agregarse en el momento de la estimación.
El método a desarrollar tendría entonces una fase más o menos automática y necesitará de algún esfuerzo específico para estimar alguna complejidad/facilidad. Afortunadamente los sistemas de gestión no nos dan excesivas sorpresas en cuanto a sus requisitos por lo cual, si la lista de escenarios normalizados está bien armada, no resulta necesaria la inclusión de muchas complejidades/facilidades. En la experiencia en Idea Factory Software como promedio por cada 40 escenarios tipo sólo se necesitó una facilidad/complejidad. Por otra parte este trabajo propone, a fin de cuentas, estimar por comparación; en nuestra lista de escenarios normalizados tratamos de tener escenarios suficientemente representativos de la realidad, pero nada quita que un escenario de la vida real, absolutamente atípico, sea comparable a uno normalizado, no por su funcionalidad, sino por su complejidad similar.
Finalmente es necesario hacer una observación importante. Sabemos que los escenarios de modificación y baja pueden ser expresados como escenarios principales del caso de uso o como alternativos del escenario de consulta. Si no se toman en cuenta complejidades y facilidades, y esos escenarios son diseñados como alternativos, entonces no estaríamos estimando las bajas y las modificaciones, dado que el método sólo contempla escenarios principales. Esto es a todas luces un despropósito. En efecto, altas, bajas, modificaciones y consultas cubren la mayor parte de cualquier sistema de gestión y deben ser incluidos sea como escenario normalizado o como complejidad standard, la exclusión de los escenarios alternativos en el método se debe principalmente a que habitualmente no se conoce ese nivel de detalle en la estimación temprana.

5.5. Cuantificación

Una vez resuelto el problema de la normalización, la cuantificación implica asignarle valores, cuya unidad podemos llamar puntos de escenario (en adelante PE), a cada escenario normalizado tipo-subtipo. Los valores deberán asignarse en función de mediciones previas realizadas, o si estas faltan, en función de entrevistas con distintos profesionales por especialidad, siendo la dispersión de resultados una medida de la confiabilidad de estos datos. Además, si se decide usarlas, deberán identificarse las distintas complejidades y facilidades adicionales que se pueden dar en cada caso y que, debidamente valorizadas, incidirán en la cantidad de "puntos de escenario" a asignar al escenario real.
La naturaleza del valor que asignemos a cada escenario dependerá de lo que queremos medir. Si queremos medir esfuerzo asignaremos valores relacionados con el tiempo de desarrollo que llevaría cada escenario base. Si queremos medir tiempo de análisis asignaremos valores relacionados con el tiempo de análisis que llevaría cada escenario base. Y así podríamos definir para un escenario tipo puntos de escenario de desarrollo, puntos de escenario de análisis, puntos de escenario de testing etc. Sin embargo este procedimiento es trabajoso, y dada la poca exactitud de la actividad de estimación en general, se justifica utilizar una actividad única y extrapolarla al resto de las actividades. Ese problema es el que llamamos "extrapolación".

5.6. Extrapolación

Como dijimos, a efectos de simplificar es conveniente entonces adoptar una actividad única para la cuantificación de los escenarios tipo y a partir de allí utilizar esa escala para las otras actividades. La práctica realizada por Idea Factory Software demostró que, dado el carácter aproximado de la estimación, esta simplificación representa suficientemente la realidad, aunque implique eventualmente tener una escala de valores distinta para alguna tecnología que se desvíe de la relación más usual entre los distintos tipos-subtipos. La actividad utilizada, y que recomendamos, fue el tiempo de codificación de cada escenario.
Es una tentación elegir el tiempo de análisis como actividad base ya que depende sólo de la metodología de análisis y no de la tecnología. En ese sentido estaríamos cerca nuevamente del concepto de tamaño. Sin embargo los valores de análisis resultan más dispersos que los de codificación por lo que podemos asumir que son menos seguros. Si tenemos en cuenta además que la codificación es la actividad que lleva mayor tiempo (relación de 3 a 1 en J2EE, por ejemplo) la elección de la codificación como actividad para cuantificar la escala resulta la más adecuada. Finalmente podemos afectar los resultados que correspondan por factores de ambiente y complejidad técnica general, pudiendo usar en este caso el ajuste propuesto por Karner [3] o por otro método probado.

5.7. Comparación de un método basado en escenarios principales vs. uno basado en casos de uso

En esta sección desarrollaremos un ejemplo que permite ver la sensibilidad de un método basado en escenarios principales con respecto al Use Case Points [3]. Tomemos para el ejemplo un caso de uso con los escenarios principales de alta, consulta y modificación (en adelante ACM) sobre una entidad simple. Consideramos que no se contempla la baja física porque la misma es lógica y es resultado de una modificación. Con el método Use Case Points tendríamos:

3 transacciones ➔ caso de uso simple ➔ Factor de peso en Use Case Points = 5.

Consideremos ahora la necesidad de hacer una baja física y entonces se agrega ese escenario al caso de uso (en adelante ACMB para el caso de uso con escenario de baja) tenemos entonces:

4 transacciones ➔caso de uso medio ➔ Factor de peso en Use Case Points = 10.

El **aumento de peso por agregar el escenario de baja es de 100%.** " Baja" es el escenario que menos esfuerzo lleva de los cuatro, pero su inclusión duplicó el peso. Esto demuestra que al método UCP le falta, al menos, suficiente granularidad para medir correctamente este tipo de diferencias.
Probemos ahora con un método basado en escenarios principales. La tabla parcial para los escenarios elegidos, para tecnologías Java y .Net (tienen iguales relaciones relativas entre tipos de escenario) es:

Tabla (parcial) de escenarios con valoración de PE para tecnologías J2EE y .NET	
Escenario - Complejidad	**Puntos de Escenario (PE)**
Escenario "Alta" – "Simple"	+10
Escenario "Consulta" – "Simple"	+22
Escenario "Modificación" – "Simple"	+6
Escenario "Baja" – "Simple"	+4

Para el caso de uso ACM el método basado en escenarios principales da :

"Alta Simple" + "Consulta Simple" + "Modificación Simple" = 10PE + 22PE + 6PE = 38 PE

Si agregamos la baja para formar el caso de uso ACMB tenemos:

"Alta Simple"+"Consulta Simple"+"Modificación Simple"+"Baja Simple"=10PE+22PE + 6PE + 4 PE = 42PE

El aumento por agregar el escenario de baja es de 10,5%, cifra que representa el aumento de esfuerzo real medido para escenarios de baja simple. Demostrando así que **el uso de escenarios principales es sensible y proporcional a los pequeños cambios.**
Veamos qué pasa con la complejidad. Consideremos ahora que la entidad sobre la que se ejecutan el alta, baja, modificación y consulta es una entidad compuesta que representa una relación de agregación (tipo cabecera-detalle). Para el caso de uso ACMB en el método Use Case Points no hay cambio evidente dado que a pesar de la mayor complejidad del caso de uso sigue siendo complejidad media porque el actor sigue teniendo 4 transacciones, es decir este tipo de cambio de complejidad no es detectado
Para el método basado en escenarios principales la tabla de los escenarios involucrados es:

Tabla (parcial) de escenarios con valoración de PE para tecnologías J2EE y .NET	
Escenario - Complejidad	**Puntos de Escenario (PE)**
Escenario "Alta" – "Cabecera – detalle"	+28
Escenario "Consulta" – "Cabecera – detalle"	+25
Escenario "Modificación" – "Cabecera – detalle"	+10
Escenario "Baja" – "Cascada"	+5

Nota: el mayor peso del escenario de “alta” con respecto al de “consulta” se debe a que el escenario tipo “alta” incluye las pantallas y navegación que luego se reutilizará en los otros escenarios. En este caso tendremos :

Escenarios Cabecera-detalle de “Alta” + “Consulta” + “Modificación” + “Baja” = 28 + 25 +10 + 5 = 68 PE

Vemos que el **aumento en complejidad fue tenido en cuenta** por el método e incrementó los puntos de escenario en un 61,9%, cifra que concuerda con las mediciones realizadas.
Podemos concluir entonces que el método Use Case Points tiene algunos problemas de sensibilidad cuantitativos (cantidad de escenarios) y cualitativos (complejidad) y que un método basado en escenarios principales resuelve esos problemas. Por otra parte sus resultados no dependen de la forma en que los escenarios son agrupados en los casos de uso.
Finalmente, y asumiendo para simplificar que no hay modificaciones de ambiente ni de complejidad técnica general, podemos obtener esfuerzo para varias actividades extrapolando:

Codificación/prueba unitaria =	0,7 h/PE x suma de PE =	0,7 h/PE x 68 PE = 47hs
Análisis =	0,24 h/PE x suma de PE =	0,24 h/PE x 68 PE = 16hs
Diseño =	0,12 h/PE x suma de PE=	0,12 h/PE x 68 PE = 8hs
Testing (planif.y ejecución) =	0,12 h/PE x suma de PE =	0,12 h/PE x 68 PE= 8hs
Corrección de defectos =	0,12 h/PE x suma de PE =	0,12 h/PE x 68 PE= 8hs

Donde los coeficientes usados para codificación y prueba unitaria, corrección de defectos y diseño se corresponden con un determinado lenguaje (Java) y arquitectura específica, y los coeficientes de testing y análisis se corresponden con un método y documentación evaluados CMM nivel 3.

6. CONCLUSIONES

Todo lo expuesto en este trabajo puede resumirse en las siguientes conclusiones: Los casos de uso no son un elemento básico apto para una estimación razonablemente fina de las actividades de desarrollo de sistemas.

[a] Los escenarios principales son un elemento básico apto para estimar las actividades de desarrollo de sistemas en general y orientados a objetos en particular.

[b] Un método para dimensionar un sistema puede basarse en una comparación entre los escenarios del sistema real a construir y una lista de escenarios normalizados. A los efectos de lograr una estimación más fina, cada tipo de escenario normalizado puede ser dividido en subtipos teniendo en cuenta algún atributo del escenario relacionado con la complejidad de su construcción o análisis.

[c] Si se necesitara mayor fineza en la estimación pueden agregarse complejidades o facilidades, normalizadas o ad hoc.

[d] El concepto de tamaño no es tan claro en la ingeniería de software como en otras disciplinas de la ingeniería.

[e] En función de la experiencia de la empresa Idea Factory Software utilizando estos principios puede afirmarse la factibilidad técnica y económica de su aplicación.

REFERENCIAS

[1] Ana Ma. Moreno Sánchez Capuchino Módulo I *Control y gestión de proyectos software -– Unidad 4: Estimación de Proyectos Software.* Carpetas de la Carrera de Postgrado en Ingeniería de Software. Imprenta del Instituto Tecnológico de Buenos Aires.

[2] Ivar Jacobson, Grady Booch, James Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de Software* Editorial Pearson Educación. ISBN: 84-7829-036-2.

[3] Gustav Karner. 2001. *Use Case Points - Resource Estimation for Objectory Projects*. Objective Systems SF AB

[4] John Smith. 1999. *The Estimation of Effort Based on Use Cases*. Rational Software White Paper.

[5] Kurt Bittner. 2001. *Why Use Cases Are Not "Functions"*. Rational Software.

[6] Alistair Cockburn**.** 2000. *Writing Effective Use Cases*. Addison-Wesley.

[7] Ministerio de Adm. Públicas. 1994. *Metodología Métrica Versión 3 – Técnicas y prácticas* - España

[8] Ivar Jacobson, Grady Booch, James Rumbaugh. 2002. *The Unified Modeling Language User Guide* - Editorial Addison – Wesley. ISBN: 0201571684.