TRABAJO DE ESPECIALIDAD EN CONTROL Y GESTIÓN DE PROYECTOS SOFTWARE

PRINCIPIOS PARA UN MÉTODO DE ESTIMACIÓN DE PROYECTOS DE SOFTWARE BASADO EN LOS ESCENARIOS PRINCIPALES

AUTOR : M. ING. JOSÉ IGNACIO CAO

DIRECTOR

DR. RAMÓN GARCÍA MARTÍNEZ (ITBA)

BUENOS AIRES, 2006

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.

Actualmente la metodología de análisis, diseño y programación orientada a objetos ha obtenido gran preponderancia en el ámbito del desarrollo de software.
Dentro de esta metodología se ha adoptado, casi universalmente, el Lenguaje Unificado de Modelado (Unified Modeling Language).
La unidad de trabajo y control en un sistema modelado en UML es el Caso de Uso y entonces debería buscarse una forma de asignar unidades de medida a los casos de uso para poder estimarlos. Sin embargo hay diferentes estilos de agrupar escenarios en casos de uso, y por otra parte la complejidad de cada escenario puede tener grandes variaciones, haciendo entonces del caso de uso una unidad no especialmente apta para la estimación.

El presente trabajo propone el uso de escenarios normalizados para la estimación de los casos de uso y por ende de los sistemas. Explica la desventaja de otros métodos utilizados y finalmente presenta un caso práctico de aplicación de los principios expuestos .

TABLA DE CONTENIDOS

CAPÍTULO 1

INTRODUCCIÓN

1.1.- LA ESTIMACIÓN EN LA GESTIÓN DE PROYECTOS

La estimación de la duración de las actividades que conforman el desarrollo de software es un tema que concierne a la gestión y control de proyectos.

En efecto, Ana Ma. Moreno Sánchez Capuchino dice « La primera tarea en la gestión de proyectos es la estimación. "[SANCHEZ CAPUCHINO, 1996]

Por otra parte el Project Management Institute (en adelante PMI) define un área de conocimiento (Knowlegde Area) dedicada a la administración del tiempo del proyecto (Project Time Management) [PMI,2000].

El PMI también define un grupo de procesos llamado "Planificación". Dentro de ese grupo de procesos, en el área de conocimiento mencionada tenemos una actividad relativa a la estimación de la duración de las actividades (6.3 Activity Duration Estimating). En ese item se define la estimación de la duración de actividades como el proceso de tomar información sobre el alcance del proyecto y los recursos, y entonces determinar las duraciones para ser usadas como información de entrada en los cronogramas.

Ana Sánchez Capuchino define 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." [SANCHEZ CAPUCHINO, 1996].
Por otra parte una definición no técnica de estimación dice que "es un conjunto aproximado de valores para algo que ha de ser hecho".

Las variables a estimar dentro del marco de un proyecto de desarrollo de software pueden ser muchas. Estas van desde la estimación de las horas necesarias para realizar tareas tales como análisis, diseño, gestión, desarrollo etc. hasta la estimación de índices tales como cantidad de defectos por unidad de medida, casos de prueba por caso de uso etc.

No es necesario enfatizar la importancia de las estimaciones en la gestión de proyectos. Cualquier cronograma de tareas pendientes implica estimaciones. Cualquier intención de calcular tiempos, recursos o costos a futuro implica estimaciones. Sin estimaciones no podría haber gestión de proyectos por que el proyecto es, como su nombre lo indica, una proyección.

En este trabajo en particular nos centraremos en llegar a sentar las bases de un método que permita, en última instancia, la estimación del esfuerzo necesario para la codificación (que

también llamaremos programación o desarrollo) del software, esto es la generación de código ejecutable o interpretable. Por otra parte al centrarnos en la codificación es fácil extrapolar conceptos e incluso valores a otras actividades directamente proporcionales a esta variable.

1.2.- BREVE DESCRIPCIÓN DEL PROBLEMA

Los problemas de la estimación son varios.

Sabemos que estimar significa, de alguna forma, predecir el futuro, actividad esta cuya incertidumbre se refleja en la frase "rango aceptable de tolerancia" de la definición dada.

Es decir, un primer inconveniente que encontramos es asignar un valor aceptable a cada elemento.

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.

Las diversas técnicas de estimación buscan entonces reducir el elemento a estimar a la valoración de unidades comunes que permitan, por agregación, cuantificar un sistema . En ese sentido se han definido puntos de función, puntos de casos de uso, clases clave etc.

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 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.

En efecto, la correcta estimación temprana de un proyecto de software es una tarea difícil o casi imposible. Esta situación es conocida y es por eso que cada vez más las empresas de desarrollo intentan vender el análisis y diseño separado de la codificación, a fin de que esta última pueda ser estimada sobre bases más firmes. En particular el Proceso Unificado de Desarrollo de Software sostiene que sólo al final de la fase de Elaboración se está en condiciones de hacer una propuesta económica firme, lo cual implicar haber consumido del 25% al 30% de los recursos del proyecto al llegar a ese punto [JACOBSON, BOOCH, RUMBAUGH, 2000].

Independientemente de lo que resulta más correcto, lo cierto es que el mercado exige cerrar un precio antes de comenzar los trabajos. En ese contexto la única información existente es un relevamiento, no muy detallado, de la funcionalidad del sistema. Esta información se utiliza para hacer la estimación del costo.

El objetivo de este trabajo es demostrar la conveniencia y factibilidad del uso de escenarios principales normalizados como elemento básico de comparación, adecuado tanto para estimación temprana como en cualquier otro punto del proceso.

1.3.- ESTRUCTURA DEL TRABAJO

En el Capítulo 2 estableceremos brevemente las características que debe tener un buen elemento básico de estimación para ser usado en la estimación temprana y que sea útil para las distintas estimaciones a lo largo del proyecto.

En el Capítulo 3 justificaremos lo poco adecuado que consideramos el caso de uso como elemento básico de estimación.

En el Capítulo 4 haremos comentarios sobre los conceptos de tamaño y esfuerzo, como sus límites en la práctica son más difusos que lo considerado habitualmente y su relación con la estimación basada en escenarios.

En el Capítulo 5 demostraremos porqué los escenarios principales son un elemento de estimación adecuado y sentaremos las bases para un método basado en los escenarios principales.

En el Capítulo 6 comentaremos un caso práctico de aplicación de los principios expuestos en el capítulo anterior por parte de la empresa argentina Idea Factory Software.

Finalmente en el Capítulo 7 presentaremos las conclusiones del trabajo.

El Anexo contiene la bibliografía citada y un glosario de términos.

CAPÍTULO 2

CARACTERÍSTICAS DE UN BUEN ELEMENTO BÁSICO DE ESTIMACIÓN

2.1 EL ELEMENTO DE ESTIMACIÓN Y LA MÉTRICA DE PRODUCTO

Si nuestro objetivo es medir el software debemos establecer algún elemento o parámetro básico de estimación a usar en una métrica.
Es decir, el elemento básico representar el software, o parte de él, y deberá ser susceptible de asignársele algún valor para llegar a un número final que expresará en forma cuantitativa la medida de ese software.

Es necesario aquí aclarar el uso del término elemento o parámetro básico de estimación. Si por ejemplo decimos que una métrica es la suma de los "puntos" de las clases clave del sistema, en esa definición las "clases clave" es lo que llamamos el elemento básico o parámetro básico de la métrica.

En ese sentido estamos buscando un buen elemento básico para armar con él una métrica para estimación. Nuestro objetivo final es medir el producto a obtener para deducir las medidas (coste, tiempo y esfuerzo) del proceso que generará el producto.

Vemos entonces que hay dos números finales que nos interesan, la medida del producto y las medidas del proceso.

La Lic. Sánchez Capuchino [SANCHEZ CAPUCHINO, 1996] define:

"*Las métricas de producto* son medidas del producto software durante cualquier fase de su desarrollo, desde los requisitos hasta la instalación.
Las métricas de producto pueden medir la complejidad del diseño, el tamaño del producto final (fuente u objeto) o el número de páginas de documentación producida.
Las métricas de proceso son medidas del proceso de desarrollo del software tales como tiempo de desarrollo total, esfuerzo en días/hombre o meses/hombre de desarrollo del producto, tipo de metodología utilizada o nivel medio de experiencia de los programadores."

La misma autora define las características que debe tener una buena métrica:

" • *Objetiva.*
• *Sencilla,* definible con precisión para que pueda ser evaluada.
• *Fácilmente obtenible* (a un coste razonable).
• *Válida,* la métrica debería medir exactamente lo que se quiere medir y no otra cosa.

• *Robusta* Debería ser relativamente insensible a cambios poco significativos en el proceso o en el producto."

En principio podría decirse que este trabajo se refiere a obtener finalmente una métrica del producto, aunque, como veremos en el capítulo 4, en software la diferencia no es tan nítida como en otras actividades.

2.2 EL ELEMENTO BÁSICO DE ESTIMACIÓN Y LA ESTIMACIÓN TEMPRANA

Nuestro elemento de estimación deberá llevar a una métrica que deberá poseer todas las características nombradas aunque, en el ámbito de la estimación temprana, cobran particular importancia las características de fácilmente obtenible y sencilla.

En efecto fácilmente obtenible implica a un coste razonable. La estimación temprana forma parte de la información de entrada para una propuesta técnico-económica. Esa propuesta puede ser aceptada o no por el eventual cliente. En caso de no ser aceptada todo trabajo involucrado en la misma es un gasto, y como tal, improductivo. Esto cobra especial importancia si se tiene en cuenta que usualmente se realizan varias propuestas que se rechazan por cada proyecto efectivamente obtenido.

Por otra parte muchas veces el tiempo de que se dispone para hacer esa estimación temprana es más bien escaso.

Otro inconveniente que existe en la estimación temprana es que la cantidad de información que se tiene sobre el sistema puede ser, en principio, bastante limitada. Nuestro elemento de estimación debe ser lo suficientemente sencilla y fácilmente obtenible como para poder ser tomada con la información que habitualmente existe en la etapa de preventa, esta información, en general, se limita al comportamiento funcional del sistema y, eventualmente, a la tecnología a utilizar.

2.3 ESTADO ACTUAL DE LA TÉCNICA - LOS PUNTOS DE FUNCIÓN

Una medida del tamaño del 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.

Los puntos de función de un sistema software se calculan teniendo en cuenta:

- Entradas al sistema
- Salidas del sistema
- Consultas
- Grupos de datos lógicos del sistema
- Grupos de datos lógicos que no son del sistema pero que el sistema usa.

Debe computarse cuántas ocurrencias de cada parámetro contiene un sistema, calificándolos según su complejidad en alta, media y baja. Cada parámetro, para una complejidad dada tiene un determinado peso, ese peso son los puntos de función asignados a ese parámetro.

Por ejemplo , una entrada de complejidad alta equivale a 6 puntos de función.

Luego de este proceso los puntos de función se ajustan de acuerdo a las características generales del sistema.

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 un inconveniente. En efecto, los puntos de función implican que uno conoce 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 al 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 con metodologías orientadas a objeto.

Si tomamos como ejemplo el Proceso Unificado de Desarrollo [JACOBSON, BOOCH, RUMBAUGH, 2000] vemos que una de las tres características principales de este proceso es estar conducido por casos de uso. Se utilizan casos de uso para hacer la especificación funcional, el análisis se compone de la realización en análisis de cada caso de uso, el diseño se compone de la realización en diseño de cada caso de uso, los desarrolladores desarrollan casos de uso, que son probados con casos de prueba derivados de los casos de uso.
Si bien la técnica de casos de uso no es privativa de las metodologías orientadas a objetos, casi todas las metodologías de este tipo se basan en esa técnica.

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.

2.4 CARACTERÍSTICAS DE UN BUEN ELEMENTO DE ESTIMACIÓN

En conclusión podemos decir que nuestro método de estimación se debería basar en un parámetro o elemento de estimación que tenga las siguientes características:

- Objetivo
- Fácilmente identificable
- Apto para ser valorado numéricamente
- Válido
- Obtenible en forma sencilla con la información existente en la etapa de preventa
- Apto para ser refinado a medida que se obtiene mayor información
- Compatible con los elementos de la metodología utilizada (para este trabajo, orientación a objetos en general).

CAPÍTULO 3

LOS CASOS DE USO COMO ELEMENTO BÁSICO INADECUADO

3.1. CASOS DE USO COMO ELEMENTO BÁSICO – MÉTODO USE CASE POINTS

Como ya dijimos el Proceso Unificado de Desarrollo [JACOBSON, BOOCH, RUMBAUGH, 2000] dice que los casos de uso son la unidad de trabajo con esa metodología. En efecto, sea que adoptemos el Proceso Unificado de Desarrollo o no, la unidad de trabajo hoy aceptada casi universalmente en un enfoque orientado a objetos es el caso de uso.

Siendo esto así, es natural considerar el caso de uso como un elemento o parámetro de 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 [KARNER,1993] 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 mi 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 [SMITH, 1999] 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 he visto un trabajo sobre el método de Use Case Points en 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 [BITTNER, 2001] 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 [COCKBURN, 2000], 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 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 transacciones o escenarios contenidos en cada caso de uso, es decir no es simplemente un problema de cantidad sino también de complejidad.

3.2 LAS CLASES DEL SISTEMA COMO ALTERNATIVA A LOS CASOS DE USO

En función de las consideraciones anteriores nos vemos tentados a abandonar el camino marcado por los casos de uso para buscar otras alternativas.

En ese sentido una unidad compatible con la orientación a objetos podría ser las clases involucradas en el sistema.
La metodología española Métrica 3 [MÉTRICA3,2001] 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 el lenguaje de programación utilizado. No intentaremos explicar el método, sólo diremos que, al igual que los escenarios, las clases clave pueden llevar asociadas funciones más o menos complejas y con más o menos métodos 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 objeto que los utilizan. En este sentido es importante que la estimación también pueda expresarse en casos de uso. Si bien afirmamos en este capítulo 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. Esto se comprenderá mejor con la lectura del capítulo 5.

CAPÍTULO 4

LA FALACIA DEL TAMAÑO INDEPENDIENTE DE LA TECNOLOGÍA

4.1 EL TAMAÑO Y EL ESFUERZO

Hasta este momento hemos hablado de estimación, pero evitamos cuidadosamente aclarar si nos referimos a estimación de tamaño o de esfuerzo.

No es posible seguir adelante sin hacer algunas aclaraciones 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 serrucho 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 etc..

Este esquema simple se emplea 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.

Nada más simple.

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.
Como ejemplo de lo que queremos exponer podemos utilizar el método de puntos de función de Albretch dado que se afirma que este método mide tamaño independiente de la tecnología.

Como dijimos en el capítulo 2, el método de Albretch asigna puntos de función a diversas funciones tales como entrada de datos, salidas y consultas, además de asignar puntos de función a los conjuntos de datos que utiliza el sistema.
El método dice que una entrada de complejidad media tiene 4 puntos de función y que a una consulta media también le corresponden 4 puntos de función.

Hasta aquí todo parece bien, pero 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 las 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 hay una explicación, que en esos proyectos Albretch encontró 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 entonces 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 puramente 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 el tamaño, en realidad 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.

Consecuentemente con lo anterior en la implementación práctica de este método realizada por Idea Factory Software y descripta en el capítulo 6 , fue necesario realizar una distinta escala de valores para alguna tecnología en donde las relaciones de esfuerzo entre las funciones variaba notablemente con respecto a las otras.

CAPÍTULO 5

LOS ESCENARIOS PRINCIPALES COMO ELEMENTO BÁSICO ADECUADO

5.1 TRES PROBLEMAS: NORMALIZACIÓN, CUANTIFICACIÓN Y EXTRAPOLACIÓN

En el capítulo 3 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 cumpla con las características descriptas en el capítulo 2, 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 cuantificación 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 ELEMENTO BÁSICO 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 elemento 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 apropiado 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 descripta en el capítulo 6 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 el capítulo 5, 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 [UML,2000] 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 comienzan a cumplir 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 casos de uso. Representan la funcionalidad del sistema, con información disponible en etapas tempranas del desarrollo y además la clasificación del sistema en escenarios no es trabajo adicional ya que de todos modos debe hacerse para un análisis orientado a objeto.

5.3 NORMALIZACIÓN

Volviendo a los problemas enunciados anteriormente (normalización, cuantificación, extrapolación), aplicados ahora a escenarios principales, comencemos por 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 cualquier escenario que encontremos en la realidad (luego se verá cómo esto es posible). Además, para una mejor clasificación relacionada con la complejidad, cada escenario de esa lista podrá tener subtipos.

En ese esquema de solución, la información que debe registrarse para cada tipo de escenario de la lista de escenarios normalizados es la siguiente:

- TIPO DE ESCENARIO
- OBJETIVO DEL ESCENARIO
- SUBTIPO
- CRITERIO DE MEDICIÓN DE COMPLEJIDAD
- FLUJO DE EVENTOS TÍPICO
- SUPUESTOS

En donde:

El **tipo de escenario** es el nombre que le vamos a dar al escenario normalizado.

El **objetivo del escenario** es de alguna forma lo que el mismo logra. Esto debe ser descripto en términos genéricos. Por ejemplo: persistir una nueva instancia de una determinada clase (objetivo de un escenario que podremos llamar Alta).

El **subtipo** establece la complejidad del escenario.
El criterio de medición de la complejidad da una pauta cuantitativa de cómo clasificar el escenario en un subtipo.

El **flujo de eventos típico** es una descripción del flujo normal de ese escenario que se incluye a fin de que al usuario del método de estimación le resulte fácil determinar en qué medida su escenario del mundo real se parece al escenario normalizado.

Una vez que se tiene una lista de escenarios “normalizados” o “básicos”, la estimación de un sistema comenzará por clasificar cada escenario del sistema real en un escenario normalizado.

Ahora bien, ¿es posible hacer una lista de escenarios normalizados tal que cualquier escenario de un sistema real pueda asimilarse a un escenario de la lista?. La respuesta es sí.

Esta respuesta no está basada en una consideración teórica sino en la experiencia descripta en el capítulo 6. En ese capítulo se explica como la empresa argentina Idea Factory Software desarrolló un método de estimación basado en los principios que exponemos. Ese método está en uso desde hace un año y medio, y hasta ahora, demostró que es posible encontrar una lista de escenarios normalizados que cubran todos los escenarios posibles, por lo menos en sistemas de gestión, que es el tipo de sistemas que hasta ahora ha desarrollado la empresa.

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.

Deberá decidirse si se adopta o no tener en cuenta complejidades y facilidades a afectar a los escenarios normalizados.

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 considere irrelevante.

Por otra parte, si se quieren usar complejidades y facilidades, es evidente que, además de la clasificación en 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 considere que es necesario agregar.

El método 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.

Es necesario aquí 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 cualquier sistema de gestión y deben ser incluidos sea como escenario normalizado o como complejidad.

5.5 CUANTIFICACIÓN

Una vez resuelto el problema de la normalización la cuantificación implica asignarle valores, que podemos llamar puntos de escenario a cada escenario tipo-subtipo.

En el caso de utilizarse un sistema de complejidades/facilidades, si cada escenario normalizado tiene un peso en "puntos de escenario" deberán identificarse además 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 base 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 medida única y extrapolarla al resto de las actividades. Ese problema es el que llamamos "extrapolación".

5.5 EXTRAPOLACIÓN

Dada una medida relacionada a una actividad (codificación por ejemplo) llamamos extrapolación a la obtención de la medida de otra actividad (análisis por ejemplo) aplicando una fórmula matemática a la medida de la primer actividad.

Por ejemplo:

Horas de análisis = f(horas de codificación)

Es decir el cálculo de las horas de análisis puede hacerse aplicando una función cuya variable sean las horas de codificación.

En términos más generales podemos definir una unidad cualquiera, que podemos llamar puntos de escenario, y que cualquier actividad sea función de ella.

Por ejemplo:

Horas de codificación = f(puntos de escenario)
Horas de análisis = g(puntos de escenario)
Horas de testing = h(puntos de escenario)

En donde f, g y h son las distintas funciones que convierten los puntos de escenario en esfuerzo de actividades específicas.

5.6 AL ENCUENTRO DEL TAMAÑO PERDIDO

Terminando de leer lo anterior nos damos cuenta de que la relación que establecemos entre puntos de escenario y los esfuerzos de las actividades es similar a la que se establece entre el método de puntos de función y el método COCOMO, y surge la tentación de considerar que entonces los puntos de escenario ¡son tamaño!.

En realidad no es así, sabemos que para asociar puntos a un escenario es necesario definir una actividad que permita comparar los distintos escenarios base, actividad que podría ser, por ejemplo, el tiempo de codificación del escenario en una determinada tecnología. En ese caso, para la tecnología en cuestión, la función de conversión de puntos de escenario en esfuerzo de desarrollo sería la función identidad (es decir Horas de codificación =puntos de escenario).

Ahora bien, si bien el tiempo de codificación de un sistema depende de la tecnología el tiempo de análisis sólo depende de la funcionalidad del sistema.
¡Parece que encontramos aquí el concepto de tamaño que habíamos perdido! ¿Qué tal si los puntos a asignar a un escenario base fueran directamente las horas de análisis que nos lleva analizar ese escenario base?
La idea parece no tener fallas, las horas de análisis serían las mismas para cualquier tecnología (asumiendo cierta normalización en el método de análisis). Esto básicamente es cierto, y podríamos tener lo más parecido que se puede llegar al concepto de tamaño.

Esta es una opción que podría explorarse, sin embargo yo no utilizo las horas de análisis como actividad básica a extrapolar.
Hay por lo menos una razón para ello, esa razón está relacionada con la robustez (ver capítulo 2) de la medida.
La relación entre tiempo de análisis y tiempo de codificación depende de la tecnología en que se va a desarrollar (asumiendo metodología de análisis fija). Hoy en día, la necesidad de funcionar en Internet que tienen los sistemas, su arquitectura de capas y su servicio

superior al de los sistemas tradicionales, resulta en relaciones análisis / codificación superiores a los que se consideraban tradicionalmente.

Programando en estándar J2EE, por ejemplo, no es raro pensar que por cada hora de análisis habrá al menos 3 de codificación (sin considerar corrección de defectos). Esto implica que un error de estimación de una hora en el análisis resultará en un error de 3 horas en el proceso de codificación. De esta forma, el error al estimar el análisis nos lleva a un error amplificado en la estimación del codificación.
La relación análisis/ codificación nos dice que la actividad de codificación es la que mayor peso tiene en la construcción de software, por lo cual es tiene mayor sentido basarse en esta última. Además, la duración de la actividad de análisis depende también de la eficiencia del usuario final para expresar y validar los requerimientos y su interpretación en el análisis. Esto hace que, en general, tengamos más y mejores (menos dispersas) mediciones de las actividades de codificación que de las de análisis.

5.7 PASOS DE UN POSIBLE MÉTODO

Entonces para realizar una estimación basada en escenarios principales lo que hay que hacer es:

- Identificar los escenarios principales.
- Para cada escenario principal

  Identificar el o los escenarios normalizados que mejor lo representan.
  Identificar las complejidades o facilidades que pudieran existir para ese escenario principal

- Sumar la cantidad de puntos de los escenarios y las complejidades/facilidades
- Hacer eventuales ajustes por condiciones ambientales o técnicas diferentes de las que se consideraron al determinar los coeficientes.
- Convertir esa cantidad de puntos en esfuerzo de construcción, análisis etc. por medio de coeficientes de extrapolación.
- Definir la duración deseada del proyecto, armar un cronograma y definir el personal afectado para lograr esa duración. En esa tarea se deben calcular las horas teniendo en cuenta las limitaciones de contratación (es decir, si necesitamos 2,8 programadores, seguramente utilizaremos 3 y entonces las horas de esa actividad serán más que las originalmente estimadas).

5.8 EJEMPLO DE USO DE ESCENARIOS NORMALIZADOS

5.8.1 Introducción

A los efectos de aclarar los principios expuestos, en especial en lo que significa homologar escenarios principales reales con escenarios normalizados, y para ejemplificar el uso de facilidades/complejidades presentamos el siguiente ejemplo.

5.8.2 Planteo del problema

Se plantea hacer la estimación de tiempo de codificación y tiempo de pruebas funcionales exclusivamente, de un programa que permita dar de alta los datos de una persona. Los datos son de identidad , domicilio y nivel de estudio. Entre los datos deberá figurar a qué Departamento está asignado. Sólo se contemplará un nivel de estudio y un Departamento por persona.
El programa deberá permitir la consulta de las personas que cumplen las condiciones de filtro de tres o cuatro atributos mostrándolos en una lista. La consulta permitirá mostrar los datos a pantalla completa de la persona elegida de la lista de los seleccionados. La pantalla de datos completos tendrá cuatro controles, uno para hacer un reporte (listado) de los datos de la persona, otro para dar la baja, otro para modificar y otro para mostrar los horarios de las actividades del Departamento a qué está asignada la persona. Los datos de los horarios de las actividades se obtendrán por medio de un webservice de un sistema existente. El alta de la persona deberá estar contemplado por un control que podrá ubicarse en la misma pantalla de consulta o como opción separada a elección del analista, a elección del diseñador. El reporte (listado) tiene los mismos filtros de la información listada que un listado ya existente desarrollado para el mismo sistema.
El programa se realizará en java cumpliendo el standard J2EE con una arquitectura ya definida. Se asume que los tiempos de preparación e instalación de ambiente de desarrollo no son considerados en este momento.

5.8.3 Identificación de los escenarios principales

Luego de haber leído la descripción del programa solicitado decidimos que todas sus funciones la incluiremos en un caso de uso que llamaremos “Administrar Persona”.
Las funciones a cubrir son la consulta de personas, la impresión de los datos de una determinada persona, la modificación y la baja de los datos de una persona. En función de esto definimos los siguientes escenarios principales:

- Ingresar Persona
- Consultar Personas
- Modificar Persona
- Eliminar Persona
- Listar Persona

Los escenarios Modificar, Eliminar y Listar podrían considerarse escenarios principales o alternativos de Consultar Persona, pero, aunque se clasifiquen como alternativos, deberán

ser considerados en la estimación por estar identificados, coincidir con escenarios normalizados y no alternativos de la lista de los escenarios normalizados. La función representada por la pantalla que muestra los horarios de las actividades del Departamento de la persona la consideramos parte del escenario Consultar Personas.

5.8.4 Identificación de escenarios normalizados y complejidades/facilidades

5.8.4.1 Escenario "Ingresar Persona"

Revisamos en una lista de escenarios normalizados definidos en una hipotética implementación de estos principios (ver item "5.8.9 Descripción de escenarios normalizados relacionados con el problema") y vemos que la descripción del flujo normal del escenario normalizado tipo "alta" cumple con los requisitos del problema expuesto.
Leemos en el subtipo "simple" para este tipo de escenario los datos de clasificación para ver si podemos encuadrar nuestro escenario real "Ingresar Persona" en el subtipo simple.

Los datos de clasificación del escenario tipo "Alta" dicen que para clasificar en subtipo simple la entidad no debe ser una composición jerárquica y debe tener menos de 20 atributos aproximadamente.
De lo que se desprende de la descripción del problema, "Persona" no parece tener relaciones de jerarquía del tipo cabecera detalle (master-detail o relación de agregación) y tiene claramente menos de 20 atributos, por lo cual lo podemos clasificar como subtipo "simple".
De la descripción puede desprenderse que posiblemente "Departamento" y "Nivel de estudio" se elijan de una lista desplegable, sin embargo esto es normal para un alta y, en la hipotética lista de complejidades/facilidades, no hay ninguna complejidad asociada a esta característica.

Entonces nuestro escenario real "Ingresar Persona" es entonces clasificado como:

- Escenario normalizado "Alta " "Simple"

5.8.4.2 Escenario "Consultar Personas"

Revisando nuestra hipotética lista de escenarios normalizados es evidente que este escenario puede clasificarse como de tipo "consulta" (ver item "5.8. 9 Descripción de escenarios normalizados relacionados con el problema")
Los datos de clasificación de subtipo son idénticos al de escenario "Alta" por lo cual podemos clasificar este escenario como "Simple".
Revisamos el flujo normal del escenario "Consulta" y vemos que existe una diferencia con lo descripto para el escenario real "Consultar Personas". En efecto, nuestro escenario normalizado no tiene en cuenta las consultas de los horarios del Departamento que contempla el escenario "Consultar Personas". Tenemos que contemplar esa diferencia con alguna de las siguientes tres acciones:

a) Determinar otro escenario normalizado que represente esa parte (nada quita que un escenario real sea representado por más de un escenario normalizado)
b) Buscar una complejidad/facilidad que represente esta pantalla de consulta extra.
c) Crear una nueva complejidad/facilidad que tome en cuenta este caso particular.

La opción a) no corresponde porque los escenarios normalizados Consulta o Listado , que son los que a fin de cuenta muestran algo, son bastante más complejos que simplemente mostrar una pantalla con datos. El escenario normalizado "Modificación" modifica datos que no es el presente caso. Revisando las complejidades/facilidades (ver item "5.8. 9 Descripción de escenarios normalizados relacionados con el problema") vemos que hay una complicación llamada "Pantalla Simple" que contempla una pantalla adicional mostrando datos. Decidimos representar la funcionalidad de mostrar los horarios de actividades del Departamento del ejemplo real utilizando entonces esa complejidad.

Vemos además que los datos de la pantalla de horarios de actividades del Departamento se leen por medio de un web service. Esto implica aprender a usar ese web service, hacer las pruebas necesarias y subsanar los inconvenientes que siempre surgen para ajustar la conexión entre dos sistemas extraños. Seguramente deberemos contemplar algún esfuerzo para eso. Nuevamente contemplamos las tres opciones a), b) y c) y vemos que existe una complejidad en la lista que se llama "Conexión con sistema externo 1" que explícitamente cubre esta situación.

Finalmente nuestro escenario real "Consultar Personas" es entonces homologado con (o representado por) :

- Escenario normalizado "Consulta" subtipo "Simple"
  +
- Complejidad "Pantalla Simple"
  +
- Complejidad "Conexión con sistema externo tipo 1"

5.8.4.3 Escenario "Modificar Persona"

Haciendo similares razonamientos que en los dos escenarios anteriores vemos que el escenario normalizado que le corresponde es el de tipo "Modificación" subtipo "Simple".

Entonces nuestro escenario real "Modificar Persona" es clasificado como:

- Escenario normalizado "Modificación" "Simple"

5.8.4.4 Escenario "Eliminar Persona"

Haciendo similares razonamientos que en los escenarios anteriores vemos que el escenario normalizado que el corresponde es el de tipo “Baja” subtipo “Simple”.

Entonces nuestro escenario real “Eliminar Persona” es clasificado como:

- Escenario normalizado “Baja“ “Simple”

5.8.4.5 Escenario “Listar Persona”

Haciendo similares razonamientos que en los escenarios anteriores vemos que el escenario normalizado que el corresponde es el de tipo “Reporte” subtipo “Simple”.

Sin embargo vemos que la descripción de la funcionalidad real nos dice que ya hay otro listado que contempla una pantalla con los mismos filtros que el listado de persona. En el escenario normalizado que corresponde a “Reporte” “Simple” (ver item “5.8. 9 Descripción de escenarios normalizados relacionados con el problema”) vemos que esa funcionalidad (pantalla de filtros) está contemplada, por lo cual habría que descontarla por medio de alguna facilidad.
Podemos inventar una facilidad especial para este caso y podríamos llamarla “reuso de pantalla con filtros simples” pero vemos que en la lista de facilidades aparece una facilidad llamada “reuso de pantalla”. Si bien parece menos especifica que la nuestra, para no salirnos mucho del método utilizaremos esa facilidad y luego veremos si estamos de acuerdo con la asignación de puntos de escenario que propone.

Entonces nuestro escenario real “Listar Persona” es homologado por :

- Escenario normalizado “Reporte“ “Simple”
  -
- Facilidad “reuso de pantalla”

5.8.5 Suma de puntos de escenario para el caso de uso “Administrar Persona”

En función de la clasificación que realizamos en el item anterior debemos ahora calcular los puntos de escenario que corresponden a los escenarios normalizados y complejidades/facilidades que utilizamos.

Para esto utilizaremos un cuadro que consigna los puntos de escenario para cada escenario normalizado y complejidades/facilidades.

El cuadro que corresponde a los escenarios y complejidades /facilidades utilizados es el siguiente:

Tabla (parcial) de escenarios y complejidades/facilidades con valoración de PE para tecnologías J2EE y .NET	
Escenario - Complejidad/Facilidad	**Puntos de Escenario (PE)**
Escenario “Alta” – “Simple”	+10
Escenario “Consulta” – “Simple”	+22
Escenario “Modificación” – “Simple”	+6
Escenario “Baja” – “Simple”	+4
Escenario “Reporte” – “Simple”	+20
Complejidad “Pantalla Simple”	+8
Complejidad “Conexión con sistema externo tipo 1”	Primera conexión : mínimo +10PE Conexión adicional =+ 3PE Si hay duda consultar líder técnico
Facilidad “Reuso de pantalla”	- 6PE ó 80% del valor de la complejidad por pantalla adicional que le corresponda

Vemos en este cuadro que el mismo tiene la valoración en PE para J2EE y .Net. Esto es porque, como habíamos dicho, la relación entre los valores de los distintos escenarios puede ser distinta para distintas tecnologías. Por ejemplo aquí vemos que una consulta “vale” 2,2 veces un alta, pero esa relación no tiene por qué ser cierta para otras tecnologías.

En función del cuadro anterior la valoración de los escenarios del problema planteado es la siguiente:

Escenario “Ingresar Persona” total = +10 PE

- Escenario normalizado “Alta ” “Simple” = +10 PE

Escenario “Consultar Personas” total = +39 PE

- Escenario normalizado “Consulta” subtipo “Simple”= +22 PE
  +
- Complejidad “Pantalla Simple” = +7 PE
  +
- Complejidad “Conexión con sistema externo tipo 1” = +10PE

Nota: se trata de la primer conexión a ese tipo de servicio que hace el equipo para la aplicación. Se consultó al líder técnico y dijo que, a falta de mayores datos se considerara el valor de planilla.

Escenario “Modificar Persona” total = +6 PE

- Escenario normalizado “Modificación” “Simple” = +6 PE

Nota: la pantalla de datos completos para modificación y las validaciones de cada atributo ya fueron realizadas en el escenario de alta, sin embargo no corresponde asignar una facilidad porque entre las precondiciones del escenario normalizado “Modificación” ya se tiene en cuenta que esa pantalla existe (dado que es el caso más usual). Si no se hiciera el alta, o por cualquier otra razón no existiera tal pantalla debería agregarse como una complicación.

Escenario “Eliminar Persona” total = +4 PE

- Escenario normalizado “Baja“ “Simple” = +4 Pe

Escenario “Listar Persona” total = +14 PE

- Escenario normalizado “Reporte“ “Simple” = + 20 PE
  -
- Facilidad “reuso de pantalla” = - 6PE

En función de lo anterior podemos calcular los puntos de escenario que representan el caso de uso “Administrar Persona”:

Escenario “Ingresar Persona” total = +10 PE
Escenario “Consultar Personas” total = +39 PE
Escenario “Modificar Persona” total = + 6 PE
Escenario “Eliminar Persona” total = + 4 PE
Escenario “Listar Persona” total =+14 PE

Total para el caso de uso “Administrar Persona” = 73 puntos de escenario

Aclaración: el proceso de clasificación y valoración de escenarios recién descripto parece largo y engorroso especialmente teniendo en cuenta que habitualmente hay decenas de casos de uso en un siestema, sin embargo, un analista con alguna práctica puede hacer la clasificación mientras lee el planteo del problema. La valoración es entonces trivial.
Con la experiencia de haber leído este item pruébese ahora releer el item “5.8.2 Planteo del Problema” y se verá que es posible una clasificación inmediata una vez que uno conoce la lista de escenarios y complejidades/facilidades.

5.8.6 Ajustes por condiciones ambientales o técnicas

En este paso deberían hacerse ajustes por condiciones ambientales o técnicas. El presente trabajo no sienta las bases, ni justifica, ningún método de ajuste para condiciones ambientales o técnicas. En una hipotética implementación del método podría hacerse una tabla de ajustes o utilizar una existente. Por ejemplo, el método de ajustes ambientales o técnicos de Use Case Point podría ser válido. Una explicación del método puede encontrarse en http://www.globaltester.com/sp7/usecasepoint.html. En este caso los puntos de escenario calculados en el item 5.8.5 representarían los "unadjusted use case points" (UUCP) utilizado como entrada a las tablas.

Para continuar con nuestro ejemplo asumiremos que no hay ajustes por condiciones ambientales o técnicas (las condiciones son "normales", es decir coinciden con las contempladas al asignar el valor a cada escenario normalizado) y que por lo tanto los puntos de escenario luego del ajuste siguen siendo 73.

5.8.7 Cálculo de esfuerzo por extrapolación

De acuerdo a lo explicado en el item "5.5 Extrapolación" utilizaremos los puntos de escenario calculados y ajustados para calcular el esfuerzo de codificación y prueba funcionla tal como se solicita en el planteo del problema.

Utilizaremos la primer forma de cálculo nombrada en el item 5.5, es decir, calcularemos el esfuerzo de codificación en función de los puntos de escenario y luego el resto de las actividades (en este caso sólo prueba funcional) en función de las horas de codificación calculadas.

Asumimos que la empresa que realizará la codificación tiene una tabla de equivalencias entre puntos de escenario y horas de codificación para la tecnología del presente problema. Las pautas para la obtención de este tipo de tablas serán explicadas en el item "6.3 Cuantificación".

Asumimos que nuestra tabla dice que el tiempo de codificación con tecnología J2EE es de 0.8 horas de codificación por cada punto de escenario.

Las horas de codificación serán entonces : 73 PE x 0,8 hs/PE = 58 hs

Las horas de prueba funcional las podemos dividir en horas de creación (diseño) de casos de prueba, horas de prueba, horas de corrección de código y repetición de la prueba para módulos afectados. En función de experiencias anteriores asumimos un solo ciclo de pruebas.

Los valores son:

Descripción de casos de prueba(hs) = 0,07 x horas de codificación = 0,07 x 58hs = **4 hs**

Ejecución de las pruebas (hs) = 0,05 x horas de codificación = 0,05 x 58hs = **3 hs**
Corrección de errores/modificaciones (hs) = 0.15 x horas de codificación = 0.15 x 58hs = **9 hs**
Repetición de pruebas (hs) = 0,03 x horas de codificación = 0,03 x 58hs = **2 hs**

El total del tiempo dedicado a pruebas funcionales y sus correcciones será: (4+3+9+2)hs = 18hs

Con esto termina la estimación quedando sólo la asignación de personal.

5.8.8 Asignación de personal (staffing)

La asignación de personal, habitualmente conocida como "staffing" no es parte de los conceptos del método expuesto. Para hacerlo simple diremos que la empresa considera que su personal , para períodos cortos, rinde 7hs por día (en períodos largos deben considerarse, enfermedad, vacaciones y otros conceptos de asistencia) por lo tanto se necesitará:

Desarrollador: (58hs de codificación + 9hs de corrección de errores)/7hs/día = **10 días hábiles.**

Diseñador de casos de prueba = 4hs/7hs/día = 0,6 día

Tester = (3hs de ejecución de pruebas + 2hs de repetición)/7hs/día = 0,4 día

Proponemos que el **diseñador y el tester** sean el mismo por lo cual tendríamos **1 día** de asignación.

En conclusión , para las actividades nombradas, se necesitarán dos semanas de desarrollador y un día de tester.

5.8. 9 Descripción de escenarios normalizados relacionados con el problema:

Se describen a continuación los escenarios y complejidades/facilidades utilizados en la resolución del problema. Esta descripción forma parte de una lista más larga que deberá ser elaborada en la implementación de un método que utilice los conceptos expuestos en este trabajo.

Escenario normalizado para tipo "alta" subtipo "simple"

Objetivo: persistir en la base de datos la información de una entidad.

Datos de clasificación: la entidad no es una composición jerárquica (tipo cabecera – detalle o master – detail) y tiene, como máximo, aproximadamente 20 atributos.

Precondiciones: entidad no existente en la base de datos

Postcondiciones: entidad existente en la base de datos

Flujo normal :
1.- El usuario acciona el control de alta (si aún no está en la pantalla de alta).
2.- El sistema presenta una pantalla con los atributos de la entidad en blanco listo para ser informados.
3.- El Usuario informa todos los atributos
4.- El Usuario acciona el control de confirmación del alta
5.- El Sistema presenta un mensaje pidiendo confirmación
6.- El Usuario acciona el control de confirmación
7.- El Sistema incorpora la información de la entidad a la base de datos y queda en el mismo estado del paso 2.

Flujos alternativos y excepciones consideradas dentro del standard :
Excepción: el usuario cancela. En cualquier momento el usuario decide cancelar la operación.
Excepción: clave duplicada. La entidad ya existe en la base de datos.
Excepción: el usuario no confirma . En el paso 6 del flujo normal el Usuario decide no confirmar.

Escenario normalizado para tipo "consulta" subtipo "simple"

Objetivo: consultar los datos de una o más entidades que cumplen ciertos criterios de selección.

Datos de clasificación: la entidad no es una composición jerárquica (tipo cabecera – detalle o master – detail) y tiene, como máximo, aproximadamente 20 atributos.

Precondiciones: entidades existentes en la base de datos. Se supone que anteriormente se desarrolló una pantalla con los datos completos de la entidad (por ejemplo en un escenario de alta)

Postcondiciones: entidades consultadas por pantalla

Flujo normal:
1.- El usuario acciona el control de consulta (si aún no está en la pantalla de consulta).
2.- El Sistema presenta una pantalla que permite informar criterios de selección de entidades (los campos de selección no exceden aproximadamente la mitad de los atributos de la entidad)
3.- El Usuario informa o no criterios de selección y acciona un control de ejecutar la selección.
4.- El Sistema, en la misma pantalla, muestra una lista con las entidades que cumplen los criterios de selección.

5.- El Usuario selecciona una entidad de la lista y acciona sobre ella.
6.- El Sistema muestra otra pantalla con los datos completos de la entidad.
7.- El Usuario consulta los datos.
8.- El Usuario acciona un control “Salir”.
9.- El Sistema vuelve a la situación del paso 4.

Flujos alternativos y excepciones consideradas dentro del standard :
Flujo alternativo: no hay entidades que se correspondan con los filtros informados.
Excepción: el usuario cancela. En cualquier momento el usuario decide cancelar la operación.

Escenario normalizado para tipo “modificación” subtipo “simple”

Objetivo: modificar la información de una entidad existente en la base de datos.

Datos de clasificación: la entidad no es una composición jerárquica (tipo cabecera – detalle o master – detail) y tiene, como máximo, aproximadamente 20 atributos.

Precondiciones: entidad existente en la base de datos. Se asume que anteriormente se desarrolló una pantalla con los datos completos de la entidad (por ejemplo en un escenario de alta o de consulta)

Postcondiciones: entidad con datos modificados

Flujo normal:
1.- El Usuario ejecuta cualquier acción que lo sitúa en la pantalla completa de la entidad (la acción puede ser ejecutar el escenario de consulta)
2.- El Usuario acciona el control de modificación
3.- El Sistema desprotege los campos modificables
4.- El Usuario modifica los datos que quiere modificar
5.- El Usuario acciona el control de confirmación de modificación
6.- El Sistema presenta un mensaje pidiendo confirmación
7.- El Usuario acciona el control de confirmación
8.- El Sistema modifica los datos de la entidad en la base de datos y queda en el mismo estado del paso 1

Flujos alternativos y excepciones consideradas dentro del standard :
Excepción: el usuario cancela. En cualquier momento el usuario decide cancelar la operación.
Excepción: el usuario no confirma . En el paso 6 del flujo normal el Usuario decide no confirmar.

Escenario normalizado para tipo “baja” subtipo “simple”

Objetivo:eliminar (física o lógicamente) la información de una entidad existente en la base de datos.

Datos de clasificación: la entidad no es una composición jerárquica (tipo cabecera – detalle o master – detail) y su eliminación no implica eliminación en cascada de otras entidades.

Precondiciones: entidad existente en la base de datos. Se asume que anteriormente se desarrolló una pantalla con los datos completos de la entidad (por ejemplo en un escenario de alta o de consulta)

Postcondiciones: entidad eliminada (física o lógicamente)

Flujo normal:
1.- El Usuario ejecuta cualquier acción que lo sitúa en la pantalla completa de la entidad (la acción puede ser ejecutar el escenario de consulta)
2.- El Usuario acciona el control de baja
3.- El Sistema presenta un mensaje pidiendo confirmación
4.- El Usuario acciona el control de confirmación
5.- El Sistema elimina los datos de la entidad de la base de datos y vuelve a la pantalla que llamó el escenario.

Flujos alternativos y excepciones consideradas dentro del standard :
Excepción: el usuario cancela. En cualquier momento el usuario decide cancelar la operación.
Excepción: el usuario no confirma . En el paso 4 del flujo normal el Usuario decide no confirmar.

Escenario normalizado para tipo "reporte" subtipo "simple"

Objetivo: listar una serie de datos seleccionados según algún criterio de selección.

Datos de clasificación: Los datos a listar (columnas o atributos) son menos de 20 y no se esperan más de 2 cortes de control
Precondiciones: datos a listar existentes en la base de datos

Postcondiciones: listado emitido por pantalla y/o impresora

Flujo normal :
1.- El usuario acciona el control de emisión de reporte (si aún no está en la pantalla de emisión).
2.- El Sistema presenta una pantalla que permite informar criterios de selección de la información a listar (los campos de selección no pasan de aproximadamente 10)
3.- El Usuario informa o no criterios de selección y acciona un control de emitir el listado.
4.- El Sistema presenta en pantalla el listado generado
5.- El Usuario mira el listado y acciona el control de imprimir.
6.- El Sistema envía el listado a una impresora o cola de impresión

7.- El Sistema vuelve a la situación del paso 2.

Flujos alternativos y excepciones consideradas dentro del standard :
Flujo alternativo: no hay datos que se correspondan con los filtros informados.
Flujo alternativo: el usuario sólo consulta por pantalla pero decide no imprimir.
Excepción: el usuario cancela. En cualquier momento el usuario decide cancelar la operación.

Facilidades y complejidades

Complejidades

"Pantalla simple" = contempla la adición de una pantalla simple (hasta 20 atributos aproximadamente) que presenta los datos y no está prevista en escenario normalizado.
Valor = +7 PE.

"Conexión con sistema externo tipo 1" = contempla las conexiones con sistemas externos por webservice o stored procedure.
Valor = mínimo 10PE por la primer conexión y 3PE por cada conexión adicional. Si la suma de los PE supera el 5% del peso del sistema es conveniente consultar con el líder técnico para que evalúe más detalladamente la complejidad.

Facilidades
"Reuso de pantalla" = contempla la reutilización de una pantalla que forma parte del escenario normalizado y cuyo reuso no forma parte del mismo.
Valor= debe estimarse en función del valor del escenario normalizado o en su defecto adoptar el 80% del valor de la complejidad por pantalla adicional que le corresponda.(Ejemplo: si es pantalla simple es el (80% de 7PE)= - 6PE. Opcional = valorar directamente 6PE

CAPÍTULO 6

IDEA FACTORY SOFTWARE – UN CASO PRÁCTICO

6.1 INTRODUCCIÓN

Se entenderá mejor todo lo expuesto en capítulos anteriores si exponemos una experiencia práctica; experiencia que, por otra parte, confirma la factibilidad de la aplicación de los principios enunciados.

La empresa Idea Factory Software es una empresa certificada CMM nivel 3. Como tal cumple con determinadas condiciones entre las que se encuentra la de tener un Software Engineering Process Group (en adelante SEPG), cuyos objetivos, entre otros es investigar y proponer las mejoras metodológicas que se crean necesarias.

A fines del 2004 el SEPG dispuso la formación de un grupo de estudio con el objetivo de determinar los métodos más adecuados para estimar. Siendo yo uno de los coordinadores del grupo, y al estar desarrollando la base de la presente tesis, propuse la aplicación de los conceptos anteriormente descriptos para el desarrollo de uno de los métodos buscados.
Es así que se desarrolló un método de estimación basado en estos principios. Este método fue utilizado con éxito durante todo el año 2005, y continúa en uso, para resolver el problema anteriormente planteado, demostrando que los principios aquí esbozados pueden perfectamente ser la base de un sistema de estimación que funcione bajo las condiciones de desarrollo del mercado real argentino actual.

Es difícil dar muchas precisiones sobre el método sin revelar algo que, de alguna manera, es un secreto comercial, sin embargo hay algunas consideraciones que podemos hacer al respecto.

6.2 NORMALIZACIÓN

En primer lugar, al proponer inicialmente un conjunto de escenarios base a utilizar la reacción del grupo de investigación fue considerar que sería altamente trabajoso hacer la clasificación de escenarios reales. Por otra parte se desconfiaba de la posibilidad de cubrir casi todos los sistemas a realizar con los escenarios base propuestos. Estas parecen ser las dos reacciones iniciales naturales ante la exposición del método.

La experiencia demostró que es posible una rápida clasificación de escenarios y que los escenarios base cubren prácticamente la funcionalidad de cualquier sistema agregando quizás unas pocas complicaciones o facilidades.
En efecto, pensemos en las funciones elementales alta, baja , modificación y consulta. Poder clasificar los escenarios de este tipo es trivial.

De alguna forma la experiencia con Function Point demuestra que es posible la identificación de estas funciones dado que alta, baja y modificación se clasifican como entrada externa, y la consulta como consulta externa.
Una vez que el grupo se convenció de la factibilidad práctica del método comenzaron los trabajos.

En primer lugar fue necesario verificar que la lista de escenarios presentados cubren cualquier sistema. El ejercicio es simple, deben tomarse las definiciones de uno o más sistemas e intentar clasificar sus funciones en los escenarios candidatos. Haciendo esto con 3 ó 4 sistemas se verificaron escenarios, se crearon nuevos o se especializaron los existentes.

Finalmente la lista inicial de escenarios tipo y sus subtipos fue refinada de tal forma que se llegó al convencimiento de que podían representar la funcionalidad de cualquier sistema.

Cabe aquí la pregunta, ¿qué tan grande resultó la lista?. Fueron finalmente 11 escenarios tipo o base. Esta lista incluye los escenarios de modificación y baja considerándolos principales.

Ahora bien, cualquier lista de escenarios tipo contendrá escenarios de alta, modificación, baja, consulta y reportes, de alguna forma es la descomposición funcional que hemos hecho desde siempre.
Un alta por ejemplo puede clasificarse en los subtipos pocos atributos, muchos atributos, y tipo cabecera detalle. Por otra parte un reporte puede clasificarse por cantidad de atributos y/o cortes de control. Esta clasificación surge no más de diez minutos después de pensar en ella. El problema está en clasificar los tipos de escenario distintos de alta, baja, modificación, consulta y reportes.

Con respecto a esos escenarios puede utilizarse el criterio que convenza a cada uno. Hay algunos escenarios definidos, como por ejemplo exportación e importación de datos. Otros pueden ser más genéricos, como procesos de comparación de tablas, lectura de tablas y escritura, conciliaciones manuales o automáticas etc.

La empresa, por otra parte adoptó, el uso de facilidades y complejidades. El criterio fue tener una lista fija de escenarios tipo y hacer los ajustes necesarios con facilidades y complejidades. En ese sentido, se identificaron facilidades y complejidades típicas, pero se dejó abierta la herramienta informática que implementa el método para que el estimador pueda agregar sus facilidades/complejidades según el caso. De todos modos los escenarios tipo cubren la mayoría de los casos (como promedio en un sistema grande por cada 50 escenarios tipo sólo se necesitó 1 facilidad/complejidad).

Por otra parte se da el caso en que un escenario de la vida real se corresponde con dos escenarios normalizados. Por ejemplo, un alta que permite imprimir los datos se clasifica como un escenario de alta más uno de reporte.

Finalmente es importante destacar que la experiencia en Idea Factory SA no sólo permite demostrar que es posible la clasificación normalizada de escenarios sino que el proceso es,

en su mayor medida, repetible. Es decir, dos estimadores expertos tienden a clasificar los escenarios de la vida real con los mismos escenarios normalizados. La condición de repetible es requisito fundamental para la adopción de cualquier método racional y en particular de la estimación.

Ante la eventualidad de que un escenario de la vida real difiera notablemente de cualquier combinación lineal de escenarios normalizados, siempre queda la posibilidad de asimilarlo a uno o más normalizados, no por su funcionalidad, sino por su similar complejidad de construcción. Esta forma de estimación se llama "por analogía" y no implica una clasificación sino una comparación.

6.3 CUANTIFICACIÓN

Una vez identificada la lista de escenarios tipo y subtipo, a cada uno de esos escenarios tipo debemos asignarles "puntos de escenario".
Lo ideal es tomar medidas de lo que se tardó en codificar cada tipo de escenario en particular y realizar estadísticas sobre esas medidas. El inconveniente es que probablemente no tengamos medidos los escenarios en un principio. En Idea Factory Software se resolvió el tema haciendo entrevistas a desarrolladores, describiendo el escenario y solicitándoles una estimación de esfuerzo en una tecnología predefinida y bien conocida por ellos.

La dispersión de los valores dados en las entrevistas nos puede dar una idea de cuán cerca estamos de la realidad, en ese sentido la experiencia en Idea Factory Software fue buena obteniéndose valores que en muchos casos aún no se modificaron.

Por otra parte fue por medio de estas entrevistas que notamos que la relación entre los distintos tipos escenarios no era constante para todas las tecnologías de desarrollo. Por ejemplo, el tiempo de codificación de J2EE era mayor que el de .Net pero sin embargo el peso relativo de cada tipo de escenario se mantenía parecido, mientras que para Oracle difería notablemente por las características de generación de código que tiene. Esta diferencia confirmó las sospechas de que el "tamaño" puede definirse para una o más tecnologías pero no necesariamente es igual para todas ellas. Esto llevó a que Idea Factory haya tenido que adoptar tres escalas distintas, una para cada una de las tecnologías involucradas (aunque en realidad para J2EE y .Net podría haberse considerado una sola).

Con respecto a los escenarios menos típicos, es evidente que, por ejemplo ante una importación de datos, nadie pueda decir cuánto se puede tardar sin tener más detalles (cantidad de datos, consistencia de los datos, etc.).
En principio puede definirse algún parámetro para clasificar en subtipos (volumen de información, cantidad de tablas receptoras, etc.) y además definirse complicaciones que sumen puntos, por ejemplo, datos no depurados.

Ahora bien, hay que tener en cuenta que altas, bajas, modificaciones, consultas y reportes suelen cubrir, al menos, el 80% de cualquier sistema de gestión, con lo cual es necesario tener bien medidos esos escenarios y tener alguna estimación para el resto de ellos aunque no pueda ser tan exacta.

Por otra parte la responsabilidad de un profesional no puede ser delegada en un método ni en un programa, en ese sentido si el profesional considera que algún escenario es particularmente complejo y significativo puede consultar el esfuerzo necesario de construcción con un Líder Técnico, sin necesidad de consultar por el resto del sistema que se ajusta al método.

Finalmente existen coeficientes que deben aplicarse por variables de ambiente o tecnológicas (por ejemplo requisitos no estables, personal part time etc.). En ese sentido puede usarse cualquier conjunto de coeficientes de ajuste existentes que se consideren adecuados, dado que los mismos son independientes de como se obtiene el esfuerzo. En Idea Factory Software se adoptaron coeficientes de corrección similares a los del método Use Case Point.

6.4 EXTRAPOLACIÓN

Si los puntos de escenario nos dan esfuerzo de construcción, cualquier otra actividad de la ingeniería de software de la cual puede asumirse una relación aproximadamente lineal con el tiempo de codificación podrá calcularse por extrapolación.

Por ejemplo, podemos decir que un punto de escenario representa 1hora hombre de codificación, 0,5 horas de análisis, 0,3 horas de diseño detallado, 0,2 horas de prueba, etc. Estos números deberán salir de la experiencia o eventualmente de entrevistas.

Nada quita por otra parte, que por medio de entrevistas y mediciones, se valoren los escenarios con puntos para cada actividad. Es decir, un escenario de alta podrá tener 8 puntos de codificación, 3 de análisis, y 2 de diseño detallado, habiendo surgido estos números de la medición de la duración (o resultado de entrevistas) para cada actividad. Hacer esto implica asumir que la relación entre dos variables, esfuerzo de análisis vs. codificación, no es constante a través de los distintos escenarios. En general semejante complicación no tiene sentido en función de las inexactitud intrínseca de la estimación.

Idea Factory S.A. utilizó coeficientes de extrapolación deducidos de los cuadros de actividades que presenta el Proceso Unificado, de literatura clásica del tema para actividades que no variaron mucho (testing) y de entrevistas con especialistas de cada especialidad. En todos los casos se trataron de obtener los coeficientes de dos fuentes distintas, luego, a medida que el método se comenzó a utilizar, se midieron y refinaron esos valores.

CAPÍTULO 7

CONCLUSIONES

7.1 CONCLUSIONES RELATIVAS A LOS CONCEPTOS EXPUESTOS

1) Los casos de uso no son un elemento básico apto para una estimación razonablemente fina de las actividades de desarrollo de sistemas.

2) Los escenarios principales son un elemento básico apto para estimar las actividades de desarrollo de sistemas en general y orientados a objeto en particular.

3) 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.
   Si se necesitara mayor fineza en la estimación pueden agregarse complejidades o facilidades, normalizadas o ad hoc.

4) El concepto de tamaño no es tan claro en la ingeniería de software como en otras disciplinas de la ingeniería.

7.2 CONCLUSIONES DE LA APLICACIÓN POR PARTE DE IDEA FACTORY SOFTWARE DE LOS CONCEPTOS EXPUESTOS EN ESTE TRABAJO.

1) La práctica demostró que los escenarios de un sistema de gestión pueden descomponerse en escenarios normalizados tomados de una lista de no más de 11 escenarios.

2) La práctica demostró que la clasificación de escenarios principales en escenarios normalizados es una actividad simple y rápida.

3) La práctica demostró que con los principios enunciados en este trabajo es posible armar un método de estimación práctico, confiable, eficiente, utilizable en distintas etapas del proyecto y repetible.

ANEXOS

REFERENCIAS

[BITTNER, 2001]
Why Use Cases Are Not "Functions"
by Kurt Bittner
Rational Software – 2001

[COCKBURN, 2000]
Writing Effective Use Cases
Alistair Cockburn
Addison-Wesley - 2000

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

[JACOBSON-BOOCH-RUMBAUGH-,2002]
The Unified Modeling Language User Guide
Ivar Jacobson, Grady Booch, James Rumbaugh
ISBN: 0201571684
Editorial Addion - Wesley - 2002

[KARNER,1993]
Use Case Points - Resource Estimation for Objectory Projects
Gustav Karner
Objective Systems SF AB

[MÉTRICA3,2001]
Metodología Métrica Versión 3 – Técnicas y prácticas
Ministerio de Administraciones Públicas - España

[PMI,2000]
A Guide to the Project Management Body of Knowledge
Project Management Institute
Newtown Square, Pennsylvania USA
2000 Edition
ISBN: 1-880410-23-0 (paperback)

[SANCHEZ CAPUCHINO, 1996]
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.

[SMITH, 1999]
The Estimation of Effort Based on Use Cases
John Smith
Rational Software White Paper - 1999

GLOSARIO

Caso de uso: descripción de un conjunto de secuencias de acciones, incluyendo variaciones, que el sistema ejecuta y que rinde un resultado de valor observable a un actor.

Elemento básico de estimación: En este contexto llamamos "elemento" a uno de los tipos de elementos que componen el modelo funcional del sistema. Por ejemplo: clases, casos de uso, escenarios etc. Elemento básico de estimación será un elemento que se usará como base en una métrica utilizada para estimar algún aspecto del sistema. Sinónimo: parámetro básico de estimación.

Escenario: una secuencia específica de acciones que representa comportamiento.

Escenario alternativo: escenario de un caso de uso que es variación de otro escenario del casos de uso.

Escenario normalizado: escenario principal tipo utilizado como elemento de clasificación/comparación con escenarios reales.

Escenario principal : escenario de un caso de uso que no es variación de otro escenario del casos de uso.

Escenario real: un escenario principal de un caso de uso del sistema que se quiere estimar.

Estimación: proceso que proporciona un valor a un conjunto de variables para la realización de un trabajo, dentro de un rango aceptable de tolerancia.

Métrica del sistema: en desarrollo de sistemas se llama métrica a una medida del software. Esa medida puede ser observada directamente (métrica primitiva) o ser un algoritmo utilizado para medir algún aspecto del sistema (métrica calculada) . Ejemplo: cantidad de líneas de código, cantidad de clases, suma de puntos de escenario de los escenarios principales, cantidad de páginas de documentación fuente multiplicadas por un coeficiente de tamaño de letra, etc.

Sistema de gestión: sistema que se utiliza para implantar procesos administrativo-contables. Por ejemplo sistemas de recursos humanos, presupuesto, control de stock, facturación, compras etc. Se usa en contraposición con sistemas técnicos o científicos que tienen un componente significativo de cálculo no trivial.

Parámetro básico de estimación: sinónimo de elemento básico de estimación.

Punto de escenario: unidad de medida que permite cuantificar (directamente o por extrapolación) en qué medida una o más actividades de la ingeniería de software son necesarias para la construcción de un escenario normalizado.