ALGUNOS ARTÍCULOS DEL BLOG DE JAVIER GARZAS, VOCAL DEL TRIBUNAL TITULAR CONVOCATORIA 2016

# Estimación software, una breve introducción

<http://www.javiergarzas.com/2011/06/breve-introduccion-estimacion-1.html>

*No es la primera vez que sale por aquí el tema de la estimación software, una de las áreas menos madura en las organizaciones de desarrollo, como comentábamos hace tiempo*[*revisando datos reales de proyectos software*](http://www.javiergarzas.com/2010/09/datos-proyectos-cmmi-e-iso-15504.html);  *por ello esta serie de post sobre “una breve introducción a la estimación software”.*

*—*

*“No se puede predecir lo que no se puede medir” (Norman Fenton)*

Comentaba McConnell (en mi opinión quien tiene los trabajos más prácticos sobre estimación software) que “una estimación software es una predicción de cuánto tiempo durará o costará un proyecto”. El propósito de una estimación software es determinar si los objetivos, los tiempos de que disponemos para realizar el proyecto, son suficientemente realistas.

Hacer una buena estimación software antes de ofertar un proyecto nos puede ayudar a detectar proyectos que no conviene abordar y que no son rentables. Aunque la realidad diga que normalmente negocio, o la parte comercial, fija inamoviblemente, y sin estimación previa, el tiempo del proyecto, esto no debería evitar las estimaciones, ya que estas nos ayudarán entonces a saber de qué tamaño es el problema en que nos hemos metido. Mejor saber al principio que es imposible hacer el proyecto en el tiempo ofertado que al final del plazo, cuando ya hay muy poco margen de maniobra.

Existen numerosos métodos de estimación software, si bien estos se pueden clasificar en dos grandes grupos: aquellos que siguen un enfoque heurístico o los que siguen un enfoque paramétrico.

**Los métodos heurísticos estimación software**

Los métodos heurísticos se basan en la experiencia, y los principales son:

**– El método basado en juicio experto.** Más comúnmente llamado “a ojo”, y que consiste básicamente en preguntar a alguien con experiencia (o sin ella) cual es en su opinión la estimación software. Y que como podéis deducir… es el método más usado. Pero que tiene el problema de que se depende mucho del experto, de la experiencia que tenga y que además tiene el riesgo de que un día el experto deje la empresa y nos quedemos sin forma de estimar.

**– El método por analogía.** Que es una importante evolución del anterior, ya que se basa en experiencias documentadas de cómo fueron los tiempos en proyectos previos. El método compara el proyecto a estimar con proyectos terminados previamente que sean similares. Aquí la importante aportación es que disponemos de un método, y de que la experiencia se va guardando en una BBDD (o más comúnmente en una hoja Excel).

**Los métodos paramétricos de estimación software**

**– COCOMO (Constructive Cost Model) II**, de Boehmn, que estima el esfuerzo (horas hombre) del proyecto. Para estimar el esfuerzo requiere previamente una estimación del tamaño (funcionalidad, líneas de código, etc., este tema lo veremos luego).

**– SLIM (Software LIfecycle Management)**, de Putnam, que de manera similar contiene un conjunto de formulas de estimación software. Estas formulas se extrajeron de estudiar grandes bases de datos de proyectos, observando cómo se comportaron las estimación software y distribuciones de esfuerzo.

Los dos anteriores sirven principalmente para obtener una estimación del esfuerzo (horas hombre de proyecto) y se basan que exista previamente un cálculo del tamaño del software a desarrollar. Para determinar el tamaño del software se utilizan principalmente dos unidades de medición: las líneas de código y los puntos función. Como las líneas de código son poco exactas, lo normal es estimar el tamaño en puntos función, y para ello existen:

**– Numerosos métodos de estimación software basados en puntos función (FPA de IFPUG, COSMIC-FFP, Puntos Casos de Uso, etc.)**, que estiman el tamaño funcional de un producto software desde los requisitos.

*“Cuando puedes medir y expresarte con números sabes realmente de lo que hablas; pero cuando no puedes medir, cuando no puedes expresarte con números, tus conocimientos son escasos y poco satisfactorios” (Lord Kelvin)*

La métrica más obvia para estimar el tamaño de un producto software es la LOC (Lines Of Code), pero esta métrica suele tener muchos problemas. Algunos problemas de utilizar LOC como métrica para estimar el tamaño son: la falta de una definición universal de qué es una línea de código (¿un conjunto de tokens? ¿lo que hay antes de un punto y coma?, etc.), su dependencia del lenguaje de desarrollo (no es lo mismo una línea en Java que en C++), la dificultad de estimar LOC en fases iníciales del desarrollo, que no está claro que un número de LOC corresponda con un número de funcionalidades desarrolladas (la misma funcionalidad se puede desarrollar con muy diferentes números de LOC), etc. Para paliar estos problemas, surgió una métrica para medir el tamaño en base a los requisitos, funcionalidad, y no en la tecnología que se va a utilizar, denominada puntos función (PF).

Los Puntos Función son “una métrica para establecer el tamaño y complejidad software en base a la cantidad de funcionalidad requerida y entregada a los usuarios” o “una función que  mide el tamaño lógico o funcional de los proyectos”.

Así como existe el “metro” como unidad de medición para longitudes, Puntos Función es “el metro” para medir el tamaño de una aplicación de software. Aunque para ser exactos, los Puntos Función no serían como el “metro”, ya que no son una medida universal, y prácticamente no hay dos empresas que midan Puntos Función de la misma manera, y hay muchos métodos de calcular los Puntos Función. Ejemplos, algunos de los métodos que más se usan (o más hemos visto nosotros en empresas) para calcular los Puntos Función.

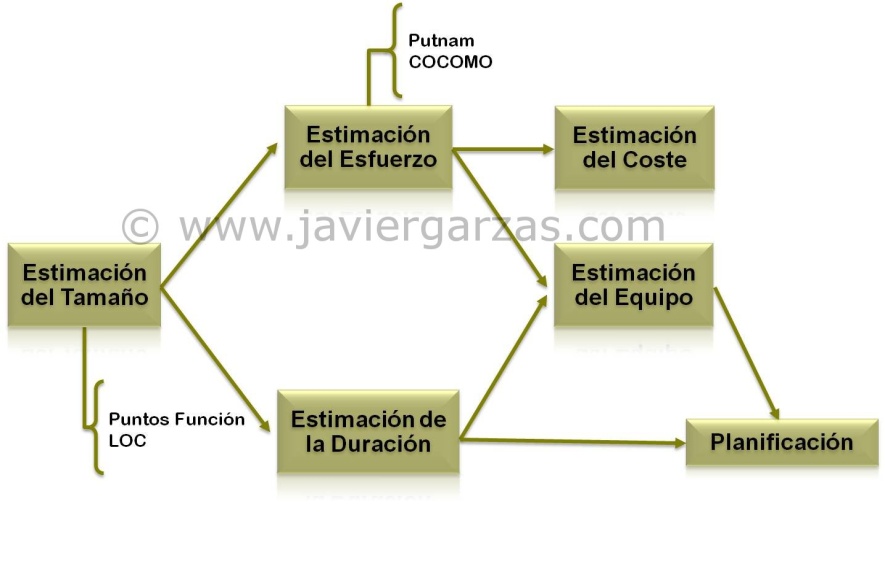
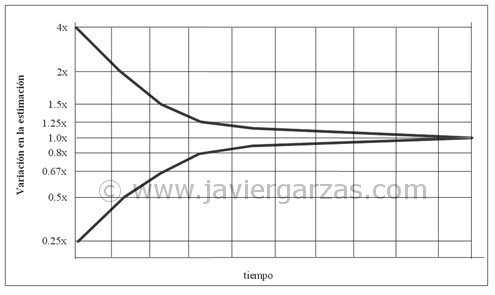
**Principales métodos para calcular Puntos Función**

**– FP Lite:** Este método deriva del método de análisis de punto función (FPA) definido por El Grupo Internacional de Usuarios de Puntos Función (IFPUG), y fue creado por la empresa “David Consulting Group”, ya que el FPA, por su tamaño y complejidad, es raramente aplicable como tal.

– **Puntos Casos de Uso (Use Case Points):** Es un método para la estimación del tamaño del software **basada en los casos de uso**. Mí método preferido, por su aplicabilidad real. Fueron introducidos por (Karner, 1993). Esté modo es similar al del cálculo de los FPA de IFPUG, pero comienza con casos de usos y actores. [Aquí tenéis una explicación mayor del método.](http://www.kybeleconsulting.com/articulos/estimacion-puntos-caso-de-uso/)

**– Early and Quick Function Points (E&QFP):**Otra técnica, que fue originalmente propuesta por Roberto Meli para, de nuevo, facilitar el cálculo de los FPA de IFPUG.  Es quizá el método más utilizado en la administración pública italiana.

Una estimación con Puntos Función, permite que la empresa tenga un método repetible de estimar, y por ello se usan sobre todo en organizaciones que externalizan mucho y quieren fijar unas reglas, un método estándar, de estimación con sus proveedores. Además los Puntos Función obtener otros diferentes indicadores, que pueden usarse para mejorar el proceso incluyendo: Número de horas de desarrolladores por Puntos Función, Número total de horas por Puntos Función, Coste por Puntos Función, etc.

* *“No se puede controlar lo que no se puede medir” (Tom De Marco)*
* Vistos en post anteriores los métodos de estimación de esfuerzo y de cálculo de tamaño, normalmente, todo proceso de estimación sigue un esquema como el de la siguiente figura:
* [](http://www.javiergarzas.com/wp-content/uploads/2011/06/estimacion.jpg)Lo normal es empezar estimando el tamaño. El tamaño suele estimarse en función de la funcionalidad que se espera que tenga el software (con PF, contando requisitos, etc.). Muchas veces las estimaciones iníciales se hacen sobre requisitos muy ambiguos (un pliego de pocas líneas) y ahí se aplican PF ligeros o, por ejemplo, se estiman los casos de uso para aplicar puntos casos de uso, y **siempre se trabaja con un error mayor de estimación** (ver en el post de mañana el cono de incertidumbre y las unidades de estimación).
* Estimado el tamaño, se estima el tiempo de proyecto (los meses de duración) y el esfuerzo (las horas hombre). Con los anteriores, ajustando el plazo en el que quiero terminar un proyecto, variando las personas que trabajarán en él, etc., se calcula el plan real (con fechas reales aproximadas) y el coste.
* Sobre cómo dado un esfuerzo total (por ejemplo 12 meses hombre) y un tiempo de proyecto (me gustaría tener el proyecto en 4 meses) se ajusta el número de personas del equipo (por ejemplo si son 12 meses hombre, se puede suponer que poniendo 3 personas el proyecto durará 4 meses) **habría que escribir otros cuatro post**. Esto es **uno de los puntos más serios y complicados, y donde más errores se comenten**. **Hay quien piensa que metiendo más gente tardará menos el proyecto, pero eso no es siempre así (**[**hacer software no es poner ladrillos en una obra**](http://www.javiergarzas.com/2011/02/diferencias-software-fabricacion-tradicional-1.html)**)**, ya que **a más gente más organización y canales de comunicación**, por lo que **la relación no es directa**. También se dice que no se puede bajar del 25% de la temporalidad estimada metiendo gente. Y en cualquier caso, ya os digo que esto es un tema de profundidad, y os recomiendo [leer al respecto todo y cualquier cosa escrita por Brooks](http://www.javiergarzas.com/2007/11/brooks-mythical-man-month.html).
* Para concluir, un conjunto de buenas prácticas útiles a la hora de trabajar con estimaciones:
* **En un proyecto real no sólo hay que estimar la construcción del software**, también hay proveedores de hardware, diseñadores gráficos, entornos de producción, etc. Otro error frecuente es obviar las tareas comunes, como vacaciones, reuniones, informes, etc., aun recuerdo aquel proyecto, de cuya empresa no quiero acordarme, en el que cuando dijimos “creemos que en la estimación hay un error porque no nos salen en el mes tantas jornadas”el comercial contesto“es que he tenido que meter los fines de semana porque no entraba en plazos el proyecto”.
* **Estimar varias veces durante el proyecto.** No basta con estimar al principio, según avanza el proyecto deberíamos reajustar la estimación. Algunos autores señalan que deberíamos estimar al menos en tres puntos: En la etapa de estudio de viabilidad, o inicio del proyecto, en la etapa de requisitos y en la etapa de diseño. Yo incluso creo que en proyectos grandes habría que estimar varias veces según avanza el desarrollo.
* **El nivel de precisión de la estimación es distinto a medida que avanza el proyecto.**Al principio del proyecto, cuando sólo se tienen requisitos, el error con el que se trabaja es mayor que cuando se estima en la fase de diseño. Una figura muy útil para entender esto es el “cono de incertidumbre”, que muestra cómo las estimaciones son más precisas según progresa el proyecto:
* [](http://www.javiergarzas.com/wp-content/uploads/2011/06/variacion-estimacion.jpg)
* **Aparte de tamaño y esfuerzo global, hay que estimar fases concretas del ciclo de vida.** Si no se tienen históricos, existen fórmulas que dicen cómo se divide en fases el esfuerzo total del proyecto, las más famosas son del ISBSG.
* **Cada empresa (y muchas veces tipo de proyecto) tiene su método de estimación.** No hay un mejor método de estimación… hay uno mejor para cada empresa, y este depende del negocio, del tamaño de la empresa, de su madurez, etc. Por ejemplo, el método de estimación para empresas que desarrollan productos no será seguramente igual que el de empresas que externalizan.
* **Utilizar varias fuentes y métodos.**Complementar siempre el método de estimación con datos de proyectos anteriores (analogía), y realizar comparativas con los mismos. Además, complementar preguntando al equipo de desarrollo.
* **Usa una unidad de estimación acorde a la fase del proyecto en la que estimas**. A la hora de proporcionar una estimación muchas veces es útil valerse de rangos, que se pueden ir ajustando conforme avanza el proyecto. Por ejemplo, “entre la primera y la segunda semana del mes de abril” o “10 meses con más menos quince días de error”. Un error muy frecuente es en la primera estimación decir el día exacto de finalización del proyecto, cuando sabemos que en una estimación tan temprana siempre hay error.
* **Último consejo.** Si os habéis leído los 4 post y habéis llegado hasta aquí, seguro que tenéis serias necesidades de implantar un método de estimación. Así que, si queréis aprender más cosas sobre estimación… [apuntaros a este curso online](http://www.kybeleconsulting.com/index.php/formacion-estimacion-software.html) ;- ).