Guía Práctica de Estimación y Medición de Proyectos Software

Julián Gómez
www.laboratorioti.com

Guía Práctica de Estimación y Medición de Proyectos Software

¿Por qué?, ¿Para qué? Y ¿Cómo?

Julián Gómez

El Laboratorio de las TI

http://www.laboratorioti.com

Título

Guía Práctica de Estimación y Medición de Proyectos Software:

¿Por qué? ¿Para qué? y ¿Cómo?

Autor

Julián Gómez

Imagen de Portada

Eukalyptus

Edición

Abril 2014

Contacto

f.julian.gomez@gmail.com

Más Información

http://www.laboratorioti.com

Sobre el Autor

Francisco Julián Gómez Bejarano

Blog: http://www.laboratorioti.com

Web: http://fjuliangomez.com

Correo: f.julian.gomez@gmail.com

Twitter: http://twitter.com/fjuliangomez

Ingeniero Superior en Informática por la Universidad de Sevilla, su proyecto fin de carrera fue *Medición del Software en Puntos Función: Desarrollo de una Herramienta*.

Certified Function Points Specialist por IFPUG (experto Especialista en Puntos Función por IFPUG) desde 2007.

Certified SNAP Practitioner por IFPUG desde 2014.

Ha participado en la traducción al español del Counting Practices Manual de IFPUG versión 4.3.1, también ha participado como miembro del equipo revisor de SNAP de IFPUG y en la traducción de éste al español y ha publicado varios artículos en la revista MetricViews.

Ha impartido formaciones / capacitaciones sobre métodos de medición funcional y otros métodos de medición en **Brasil, Chile, Colombia, España y Perú.**

Desde 2006 ha desarrollado su actividad laboral en **Servicios de Estimación, Medición y Valoración de Proyectos Software**, Gestión de la Productividad, Auditoria de Valoraciones y control de Proveedores.

Blogger en el Laboratorio de las TI (http://www.laboratorioti.com) Blog dedicado a la Gestión de Proyectos, Innovación, Estimaciones y Mediciones del Software, entre otros temas.

Prólogo

Después de llevar prácticamente un año y medio escribiendo una media de casi 3 artículos por semana en **El Laboratorio de las TI** (http://www.laboratorioti.com), uno se da cuenta de la gran cantidad de información que ha sido capaz de generar y de la necesidad de agruparla y darle cierta cohesión.

Este libro pretende recoger todos los artículos que he ido escribiendo en **El Laboratorio de las TI** sobre Estimaciones y Mediciones, corrigiéndolos, ampliándolos y añadiendo algunos temas que o bien por extensión o bien por complejidad, no son sencillos de tratar en los artículos del día a día.

Este libro justifica por qué debemos medir el software, nos muestra que podemos conseguir con la medición del software y recoge los métodos con los cuales podemos medirlo. El conocimiento de un mayor número de técnicas de medición nos dará un mejor criterio a la hora de elegir la más adecuada para nuestro proyecto.

Esta guía incluye algunos métodos que por primera vez se describen en español. Este ha sido otro de los motivos importantes a la hora de escribir sobre estos temas. Por ejemplo el método de SiFP (Simple Function Points) o el recientemente inaugurado Software Non-functional Assessment Process (SNAP), así como el método FiSMA, COSMIC, MK-II, etc.

Espero que resulte de tu interés y recuerda que para cualquier duda o pregunta que te surja puedes ponerte en contacto conmigo a través del blog **El Laboratorio de las TI** (http://www.laboratorioti.com).

No es un punto y final, es un punto y seguido donde comenzar.

Seguimos en contacto en la red.

Agradecimientos

Esta obra no podría haberse realizado sin la colaboración de una serie de personas maravillosas y, como siempre me han dicho mis padres, de bien nacidos es ser agradecido.

A **Mari Carmen**, por ser como es y por estar siempre a mi lado apoyándome.

A **mis Padres** por estar siempre ahí.

A **Julián** y a **Daniela**, por convertir cada momento en único.

Y a todos los seguidores de **El Laboratorio de las TI** (http://www.laboratorioti.com) por su continuo apoyo y soporte, que me sirven de estímulo para dar lo mejor y realizar cada vez mejores proyectos.

INDICE

Capítulo 1 La Medición del Software

¿Por qué se debe Medir el Software?

Existen muchas frases entorno a la justificación o entorno a aclarar la gran importancia que tiene el proceso de medición dentro de cualquier desarrollo y, por supuesto, dentro del desarrollo de Software.

De todas ellas, una de las que me parece más explícita es la que se incluye en *Ingeniería del Software: un enfoque práctico* de *Pressman*:

"Si no se mide, no hay forma real de determinar si se está mejorando. Y si no se está mejorando se está perdido"

Lo primero que necesitamos de un determinado proceso o de un determinado producto que estemos desarrollando, etc., es poderlo conocer, saber cómo es, cuáles son sus características. Todo debe definirse para poder dotarlo de características que lo hagan identificable frente a otros procesos y que nos permitan posicionarlo en un determinado orden con respecto a ellos.

La definición de un determinado proceso o producto, es el primer punto que nos aporta una métrica, estableciendo una línea base.

Esta línea base aportada es la que nos permite un segundo punto también importante, **la comparación entre procesos y/o productos**.

Aquí tenemos dos vertientes:

- la comparación entre productos distintos o procesos distintos para determinar de una forma objetiva cuál de ellos es el mejor o el más adecuado para nuestras circunstancias

- la comparación de un proceso con la evolución de sí mismo para poder determinar cuál es el grado de mejoría que se ha alcanzado a la

hora de implementar una serie de mejoras y acciones evolutivas/correctivas.

La medición nos permite una comparación objetiva de productos y procesos.

COMPARANDO FASES DEL DESARROLLO DE UN PROYECTO

Si esta segunda comparación la aplicamos dentro de las fases del desarrollo de un determinado producto, es decir si nos adelantamos a tomar medidas antes de que el producto final esté desarrollado, nos permitirá realizar un seguimiento y control del mismo.

Este seguimiento y control nos da una herramienta para observar el grado de desviación que se va produciendo en cada una de las fases realizadas y nos permitirá acometer medidas correctoras si fuera necesario y así poder hacer converger los valores tomados con los valores previstos.

La medición **nos permite el seguimiento y control de los desarrollos de los productos y de la evolución de los procesos**.

La medición **nos permite evaluar que producto o proceso es mejor**, en nuestras circunstancias.

ESTIMANDO PARA PLANIFICAR

Otro punto crucial, dentro del desarrollo de Software, es la realización de predicciones dentro del punto de vista de la planificación. A la hora de realizar una planificación estamos realizando una predicción de la duración, del esfuerzo, del tamaño, etc. de un desarrollo.

Para poder realizar estas predicciones de forma objetiva y precisa, necesitamos tener un conocimiento de cómo es nuestro proceso (definición), como se comporta (seguimiento y control) para poder así establecer con seguridad los patrones que seguirá. Por tanto, la medición nos permite **predecir el comportamiento y la evolución del desarrollo**.

¿QUÉ NOS PERMITE LA MEDICIÓN?

Resumiendo la medición nos permite con respecto al propio producto/proceso:

- **Su definición.** Conocer cómo es.
- **Su seguimiento y control.** Dirigir su evolución y actuar frente a desviaciones con respecto a la planificación.
- **Planificar.** Predecir resultados en función del conocimiento y del seguimiento y control.
- **Evaluar.** Nos permite poder evaluar si hemos conseguido los objetivos propuestos.
- **Mejorar.** Con la definición, evaluación y el seguimiento y control todo ello nos permite mejorar.

Con respecto a otros productos/procesos:

- **Establecimiento de una línea base** para comparar con otros procesos.
- **Evaluación de la productividad** del uso de nuevos procedimientos y herramientas, por tanto también supone una ayuda a la justificación del uso de nuevas herramientas y/o procedimientos.

¿Para qué medir?

A través de ejemplos reales de nuestra vida identificamos situaciones en las que nos dejamos llevar por nuestro instinto y situaciones en las que parece imprescindible tener un dato objetivo que nos ayude a mejorar y a evolucionar.

Así que tras analizar los comentarios y nuestras propias opiniones, la conclusión a la que llegamos, es que hay que tener en cuenta tanto las sensaciones como las mediciones

¿CUÁLES SON NUESTROS OBJETIVOS?

Una frase sobre la que me gusta desarrollar algunas presentaciones es:

"Si no puedes medirlo, no puedes gestionarlo"

Pero antes de gestionar o administrar cualquier elemento de nuestro entorno debemos identificar nuestras necesidades y establecer unos objetivos.

En esta época de crisis que nos toca vivir, a todos nosotros, tanto particulares como empresas, nos interesa controlar los costes y solemos tomar la decisión de gastar lo menos posible y ahorrar dinero. El objetivo parece estar identificado con claridad, pero la forma de medirlo suele centrarse en la salida y entrada de dinero en la caja:

El año pasado me gasté X y este año me he gastado X – 10%.

¿Estamos seguros de que se ha conseguido el objetivo? ¿Es correcto el proceso de medición que se ha seguido?

Nuestro objetivo es optimizar el gasto, no simplemente gastar menos.

EN EL GASTO NO SÓLO INFLUYE EL COSTE

¿Cuántas veces escuchamos la frase *"lo barato sale caro"?* Podemos comprar un producto de menor precio creyendo que así estamos ahorrando, pero finalmente ese producto deja de ser útil a los 6 meses y otro producto que era un

poco más caro nos habría durado 12 meses.

Para **optimizar los gastos** no es suficiente tener en cuenta una única variable (coste), debemos analizar otros elementos, como pueden ser la duración, calidad, usabilidad…

Desde casi cualquier perspectiva de nuestra vida, el objetivo de optimización de gastos puede lograrse estableciendo mejoras en:

- **Productividad**: haciendo más con el mismo esfuerzo.
- **Calidad**: reduciendo incidencias desde las primeras fases del ciclo de vida de los productos y evitando re trabajos.
- **Costes**: análisis de viabilidad de los proyectos, estudiando desde el inicio, cuál es la mejor opción: hacer o comprar.

Mis experiencias relacionadas con intentar optimizar gastos en función únicamente del coste son negativas. El cliente gasta menos a corto plazo, pero más a largo plazo y obtiene productos de menor calidad. El proveedor intenta mejorar su rentabilidad poniendo recursos de menor experiencia, ajustando salarios y aumentando el número de horas de las jornadas de trabajo.

Ninguno ve cumplidos sus objetivos y la relación suele terminar con ruptura y problemas sin resolver.

Pero **hay formas de hacer bien las cosas y lograr los objetivos que nos planteamos**, sobre todo si conseguimos trabajar todos en la misma dirección, clientes y proveedores. Hay que buscar el equilibrio entre las mejoras de productividad, mejoras de la calidad y ahorro de costes.

Este artículo ha sido escrito por Pablo Soneira. Desde aquí quiero

agradecerle su colaboración.

Reglas de oro para una estimación satisfactoria

Vamos a ver 10 reglas de oro sobre cómo debe ser una estimación. Diez reglas que nos van a permitir contar con toda la potencia y todas las ventajas que nos proporciona la estimación en la gestión de proyectos y, además, conseguiremos anular sus inconvenientes.

Una estimación debe ser:

1. **Repetible.**

Debemos poder realizarla en varios puntos y en diferentes momentos del desarrollo de un proyecto para que podamos comparar los resultados obtenidos en cada uno de ellos y extraer conclusiones.

2. **Realizada por diferentes métodos.**

La utilización de diferentes métodos nos permite una visión comparativa de las estimaciones y una mayor precisión sobre el resultado obtenido. Nos permiten detectar desviaciones propias de cada método y paliarlas de forma adecuada.

3. **Documentadas.**

Todas las asunciones que tomemos en una medición, los resultados que obtengamos, las medidas que tomemos, debemos documentarlas para poder consultar la información en el futuro y conocer la base del trabajo que realizamos. Si no lo hacemos así, a futuro podemos estar comparando peras con manzanas.

4. **Documentadas con suficiente detalle.**

Además de documentar, debemos documentarlas con el detalle suficiente para que nos permitan comparar con otros proyectos, con otros equipos, con otras empresas, etc.

5. **Una tarea más del Proyecto.**

Las estimaciones forman parte de las tareas de un proyecto. Las estimaciones no deben sentirse como una tarea separada, aparte, que esté expuestas a ser retrasadas para potenciar la consecución de hitos, los entregables o el cierre del proyecto: si perdemos la estimación, perdemos el conocimiento de la situación en la que estamos.

6. **Responsabilidad de todos.**

Cuando el responsable directo no está presente, el resto de miembros del equipo no asumen como propia la responsabilidad de seguir manteniendo las estimaciones y esto va en detrimento de la propia estimación.

7. **Parte de un histórico de estimaciones de los proyectos de la compañía.**

Permite el aprendizaje de la compañía.
Con ello evitamos:

a. Deficiencias en el saber cómo hacer las cosas mejor
b. Perder oportunidades o posibilidades de aprendizaje
c. Sufrir una crisis

Y conseguimos

a. Oportunidades de mejorar los procesos
b. Avanzar

8. **Actualizada cuando cambia la situación.**

No debemos esperar simplemente al momento en el cual esté planificado sino que en cuanto la situación cambie, varíen las condiciones que supusimos, debemos realizar una nueva estimación para conocer el nuevo resultado.

9. **Completada con los esfuerzos de estimación, de aseguramiento de la calidad y de gestión de proyecto.**

Muchas veces, inconscientemente, se olvida contemplar estos trabajos que consumen esfuerzo. Hay que medirlos y tenerlos en cuenta en todo

momento para no llevarnos sorpresas desagradables.

10. Validada por terceros.

La estimación debe ser revisada por terceros en cuanto a su completitud y a su precisión para asegurarnos de que sea con calidad.

Desventajas de las Métricas Técnicas frente a las Métricas Funcionales

Los aspectos técnicos del software también llamados *medidas a posteriori* o *medidas sintácticas*, son las medidas que se encargan de contabilizar **componentes físicos del software** tales como las líneas de código generadas, número de programas creados, número de ficheros fuente, número de páginas de documentación generada, etc.

Los aspectos funcionales del software o también llamados *medidas a priori* o *medidas semánticas*, son medidas enfocadas a contabilizar **componentes funcionales del software**, como por ejemplo los accesos a Entidades, las entradas de datos, las salidas de datos, etc.

La medidas físicas, y las métricas derivadas de ellas, se caracterizan sobre todo porque **son unas medidas muy exactas y bastante objetivas**:

El número de líneas de código de un programa es el que es, no es interpretable, aunque también tenemos que pagar un precio por esa exactitud y es que debemos esperar a que el software esté construido para poder realizar la medición de las mismas.

En un primer momento pueden parecer, por ello, unas medidas muy útiles a la hora de describir el software pero si ahondamos en su análisis encontramos una **serie de desventajas** que puede hacer contraproducente su uso:

1. **Medidas no uniformes, dificultad en estimar.**

   Como acabamos de ver, al ser medidas a posteriori, se obtiene su valor exacto al finalizar el proyecto pero es bastante difícil poder realizar una previsión de cómo irá evolucionando dicho valor conforme vaya avanzando el desarrollo del mismo.
   De hecho, hay fases de avance del proyecto que pueden no estar asociadas de ninguna forma con los parámetros físicos y, por tanto, no nos puedan servir para estimar el esfuerzo de dichas fases. Por ejemplo:

   La toma de requisitos afecta al desarrollo final del proyecto pero no se

puede estimar un número de líneas de código preciso para el control

La estimación, en estos casos, se debe basar más en una **base histórica de proyectos** dentro de la propia organización. Así podremos contrastar las características del proyecto, con otros ya realizados y que éstos nos sirvan de guía, junto con la experiencia del propio analista, para la confección de estimaciones y previsiones.

2. **Definición de la medida imprecisa.**

Como comentábamos anteriormente, el valor de la medida es exacto y es objetivo. Es una de las ventajas que poseen estas medidas pero aunque su valor, una vez establecido, es exacto, la propia definición de la medida puede no ser exacta. Por ejemplo:

Cuando definimos una medida como líneas de código fuente sin añadir más, nos puede llevar a que comparemos dos sistemas en los cuales en uno estemos contando las líneas de código fuente en un lenguaje de alto nivel y en otro estemos contando las líneas de código máquina generadas por el compilador.

También que en uno estemos contando las líneas generadas automáticamente por la arquitectura de dicho sistema y en otro las estemos obviando porque no forman parte de nuestro desarrollo

Por tanto, un problema que nos encontramos es que la definición de la medida en sí debe constar de algo más que su propio nombre porque si no, puede dar errores y que estemos comparando medidas diferentes.

3. **Medidas de naturaleza técnica.**

Estas medidas, en cuanto que son relativas a características físicas del software que se está desarrollando, implican, obviamente, que los agentes implicados en su análisis y que necesiten revisar sus valores, deban conocer el software y sus características.

Esto hace que todo aquel que no tenga conocimientos técnicos de

desarrollo de software, no pueda interpretar correctamente los distintos valores obtenidos y, por tanto, estas medidas no nos permiten aportar información al usuario final de la aplicación, que normalmente desconoce el desarrollo de software, y lo que es peor, el usuario final de la aplicación no nos puede aportar su experiencia y conocimientos del entorno y de los procesos de negocio de la aplicación para contrastar las estimaciones realizadas.

4. **Dependencia del entorno tecnológico.**

Al medir aspectos físicos del software, estas medidas tienen una dependencia extrema con respecto a las herramientas y lenguajes de programación que se hayan utilizado en el proceso de diseño y codificación del mismo.

Una vez realizado el análisis de un proyecto, podremos optar por distintos lenguajes/entornos en los cuales implementarlo y los valores físicos obtenidos para cada uno de ellos pueden variar mucho. Por ejemplo:

Si realizamos el análisis y diseño de un sencillo editor de texto y luego realizamos dos implementaciones del mismo, una de ellas en Java y otra en Visual C++, ambas implementaciones tendrán un número diferente de líneas de código fuente

Esto nos hace ligar nuestra medición a la tecnología sobre la cual estemos midiendo.

5. **Dependencia del entorno metodológico.**

La realización de los desarrollos de software bajo una metodología definida de diseño y programación, siguiendo unas pautas de estructuración y creación de clases, módulos y/o paquetes, con una nomenclatura determinada, siguiendo unas directrices para la mejora del mantenimiento del código en general, pueden hacer que un mismo proyecto implementado siguiendo las directrices establecidas o implementado sin seguirlas, tengan valores completamente diferentes en dichas medidas.

6. **Dependencia del entorno humano.**

Como acabamos de indicar en el punto anterior, el seguimiento de una metodología se traslada directamente a las medidas que podemos obtener de un proyecto en concreto. Así mismo, la capacidad de los desarrolladores, del entorno humano del proyecto, también tiene una alta influencia en el desarrollo del mismo, ya sea por la pericia/impericia de los mismos, por los conocimientos que hayan adquirido o por el desconocimiento que posean, etc.

Ventajas de las Métricas Funcionales frente a las Métricas Técnicas

Las medidas funcionales, y las métricas derivadas de ellas, tienen como principal característica que se toman de elementos del negocio, no de elementos físicos del software, siendo estos elementos de negocio entendibles por el usuario final de la aplicación. Como pueden ser entendidos por el usuario final, ellos mismos pueden revisarlos junto con los analistas del proyecto.

Como ejemplo de métricas funcionales tenemos los procesos de entrada y de salida, las entidades de datos, etc.

Utilizando métricas funcionales del software obtenemos una serie de ventajas que no teníamos con la medición de aspectos físicos del mismo, como son las siguientes:

1. **Mejora del entendimiento del usuario.**

Como hemos comentado más arriba, los elementos que se miden son entendidos y conocidos por el usuario de la aplicación porque pertenecen al dominio del negocio y no al dominio del desarrollo de software.

Mejoramos, por tanto, la comprensión de la medida que se le está aportando al usuario y conseguimos que él mismo pueda aportar su conocimiento del negocio a mejorarla y validarla.

2. **Facilidad de estimación.**

La realización de la medición de un proyecto software con respecto a aspectos funcionales, se puede realizar desde etapas muy tempranas del desarrollo. Prácticamente desde la toma de requisitos se puede realizar una primera estimación del mismo, complementándola con una medición en el análisis y con otra al final del proyecto.

Cada una de estas mediciones pueden compararse entre sí para obtener cómo ha ido evolucionando la ejecución del proyecto e ir tomando acciones de mejora si se detectasen desviaciones con respecto a lo

planificado.

También nos permiten tener una idea de cómo ha ido evolucionando la definición de los requerimientos del usuario, si los tenía claros desde el principio o han ido evolucionando hasta el final del desarrollo.

3. **No dependen del entorno tecnológico.**

Al ser medidas independientes de la implementación, las métricas funcionales no varían según el entorno tecnológico final. Por ejemplo:

Un mismo Proyecto implementado en Java y Visual C++ arrojará el mismo valor de Entidades creadas.

4. **No dependen del entorno metodológico.**

Al igual que en el caso anterior, las métricas funcionales son independientes de la metodología seguida ya que lo que se mide es el negocio y no cómo se implemente éste.

5. **No dependen del entorno humano.**

El conocimiento/desconocimiento o la pericia/impericia, a la hora de realizar la implementación del sistema, no afectan a las funciones de negocio que se deben desarrollar y, por tanto, no afectan a la funcionalidad concretada con el usuario.

6. **Benchmarking.**

Estas medidas nos permiten comparar productos del mercado independientemente de la tecnología que implementen cada uno de ellos. Podemos optar entre una herramienta u otra porque una nos ofrece más funcionalidad que la otra, desde un punto de vista objetivo con las métricas funcionales que utilicemos.

Debilidades de las Métricas Funcionales

Aunque las medidas funcionales aportan un mayor conocimiento del sistema evitando los inconvenientes que tienen las medidas técnicas y permitiéndonos realizar previsiones con mayor control de las mismas, también tienen una serie de debilidades que es necesario conocer.

Conocer estas debilidades nos permite evitarlas desde un principio o paliarlas para que su influencia sea nula o muy pequeña:

1. **Objetividad.**

Una de las principales debilidades de las medidas funcionales es que aunque la definición de la medida es clara, la medición de los elementos que la cumplen no siempre es igual de clara. Sucede al contrario que lo que comentábamos en las medidas técnicas donde la medida es exacta y la ambigüedad viene en la definición.

Veámoslo con un ejemplo:

Varios analistas pueden tener claro el concepto de medir las entidades de un determinado proyecto pero a la hora de realizar la medición cada uno interpreta que el sistema tiene más o menos entidades en función de sus propios criterios, experiencias, conocimientos, etc.

Esto se palia en diversas metodologías funcionales, como en los puntos función, estableciendo una serie de reglas que permitan a todos los analistas estar alineados a la hora de realizar las mediciones funcionales y estableciendo una certificación sobre sus conocimientos en ellas:

Dos expertos en Puntos Función no varían más de entre un 7% – 10% en sus valoraciones sobre un mismo proyecto partiendo de la misma información.

2. **Necesidad de dedicación especial al margen del desarrollo.**

En general, los aspectos funcionales del software deben ser medidos por

expertos, aunque se está trabajando desde diversos grupos en realizar automatismos que permitan generar una medición sin intervención de analistas, ahora mismo todavía no es posible. Por ello, hay que dedicar un esfuerzo al margen del esfuerzo del desarrollo propiamente dicho, para obtener las medidas funcionales del Proyecto que se esté desarrollando.

Este esfuerzo extra no suele ser muy elevado en comparación con el esfuerzo del proyecto en sí pero si hay que tenerlo en cuenta para que no se convierta en una debilidad influyente.

3. Calidad de la documentación.

Para poder realizar estas medidas dependemos totalmente de la documentación generada para los proyectos. Si la documentación no es de calidad o no existe, no podremos realizar una medición adecuada y tendremos que entrar en el terreno de las estimaciones con mayor o menor fundamento.

En cierta medida se puede paliar esta situación acudiendo a usuarios expertos que conozcan el sistema y que nos puedan transmitir el conocimiento que falta en la documentación.

Esto nos reporta una ventaja adicional y es que el usuario se implica en la medición, es partícipe de ella y la comprende.

Un Ejemplo de Métricas Funcionales vs Técnicas

Un amigo me comentó que no entendía bien que aportaban las métricas funcionales con respecto a las métricas técnicas y me pidió que le mostrara un ejemplo para que le quedara más clara la diferencia entre ambas métricas.

Entonces le dije que imaginara que quería comprarse una casa y que para ello acudía a dos agencias inmobiliarias diferentes…

La Agencia A. Cuando llega a la agencia A les indica que quiere comprar un piso/apartamento de unos 100 m^2 en una buena zona. La agencia le comenta que tiene un apartamento muy bueno que seguro que le gustará mucho. Le muestran las siguientes características:

- El piso es un ático en la calle de las Flores nº 7.
- Tiene 100m^2
- 95.000 ladrillos
- 150 sacos de cemento Cemex de primera calidad de 50Kg de peso.
- 50 m^3 de arena.
- 90m de escayola.
- 10 sacos de pegamento para azulejos de 15Kg de peso.
- 10.000 azulejos de primera calidad.
- 100 Kg de pintura blanca mate de primera calidad.
- 70m^2 de parquet (tarima flotante) de la mejor calidad.
- 50m^2 de madera.
- El precio de la casa es de 100.000€

La Agencia B. Cuando llega a la agencia B les indica también que quiere comprar un piso/apartamento de unos 100 m2 en una buena zona. La agencia le comenta que tiene uno muy bueno que seguro que le gustará mucho. Le muestran las siguientes características:

- El piso es un ático en la calle de las Flores nº 7.
- Tiene 100 m^2

- Tiene 3 habitaciones, 1 salón (living room) de 22m^2, 1 cocina y 2 baños (toilettes).
- Los cuartos de baño (toilettes) son completos incluyendo uno dentro de la habitación de matrimonio con una bañera y otro con una ducha.
- El suelo está totalmente realizado en parquet (tarima flotante) de la mejor calidad.
- Cada habitación tiene armarios empotrados hechos en madera de dos cuerpos de tamaño.
- Todo el piso está pintado en perlita blanca con pintura mate de primera calidad.
- Todas las molduras de escayola de los techos están colocadas.
- La casa es muy luminosa y tiene unas vistas muy bonitas a un parque cercano.
- En la vecindad hay supermercados, colegios y un centro sanitario.
- Hay estaciones de transporte público de metro y bus a 150 metros de distancia.
- El precio de la casa es de 100.000€

Aquí le dije a mi amigo que el piso/apartamento era el mismo y que ambas agencias habían sido exactas en las características de la vivienda. Él extrañado me comentó que no podía ser porque las características no coincidían unas con otras.

En este punto le dije que la diferencia radicaba en las métricas que utilizaban, en que cada agencia miraba la casa desde un punto de vista diferente:

- la **Agencia A** estaba dando las **características técnicas**
- la **Agencia B** estaba dando las **características funcionales**

Capítulo 2 Estimación de Proyectos

Estimación de Proyectos vs Estimación de Software

Siempre que estamos en el entorno de proyectos de software, es difícil hacer una diferenciación clara entre métodos o técnicas para estimación de proyectos y métodos o técnicas para estimación del software.

Alguien podría pensar que la diferencia entre ambos es el tipo de tareas estimadas en cada método pero no siempre es tan determinante porque puede haber métodos que permitan ambas opciones, por ejemplo:

En los Puntos Función podría pensarse que sólo se mide el tamaño del software a través de los requisitos funcionales pero si al final aplicamos un factor de productividad para obtener el esfuerzo asociado, calculado a partir de todas las tareas de gestión que se incluyen en un proyecto, el resultado es que tenemos una medida de estimación de proyectos completos y no sólo de producto software.

Por ello, el criterio que he utilizado para separar entre ambos no tiene porque ser el único pero creo que es el que mejor se ajusta con la intención principal de cada uno de ellos.

Estimación de 3 Puntos

La *Estimación de 3 Puntos, o Three-Point estimation*, es uno de los métodos utilizados en la Gestión de Proyectos cuando no se dispone de mucha información y debemos obtener la estimación de la duración o del coste de un Proyecto desde el punto de vista del Jefe de Proyecto (Project Manager).

En esta técnica, se obtiene un **valor estimado** y una **desviación típica** (como varían los valores con respecto al estimado) a partir del cálculo de los 3 valores siguientes:

- **Pesimista (Pessimistic – P).** Que sería el coste o duración del proyecto en el peor caso.
- **Más Probable (Most likely – Ml).** Que sería el caso más esperado de coste o duración del proyecto.
- **Optimista (Optimistic – O).** Que sería el coste o duración del proyecto mejor que se pudiera dar.

Por ejemplo:

Podemos tener un proyecto que sin ningún impedimento grave durará 5 semanas.

Eso sí, si tenemos problemas con el desarrollo en los navegadores podríamos ver alargada la construcción del software hasta las 10 semanas y, en cambio, si esta parte logramos elaborarla rápido, podemos superponer tareas y concluir el proyecto en 4 semanas.

Por tanto los 3 puntos de la estimación serían:

- *P = 10 semanas.*
- *Ml = 5 semanas.*
- *O = 4 semanas.*

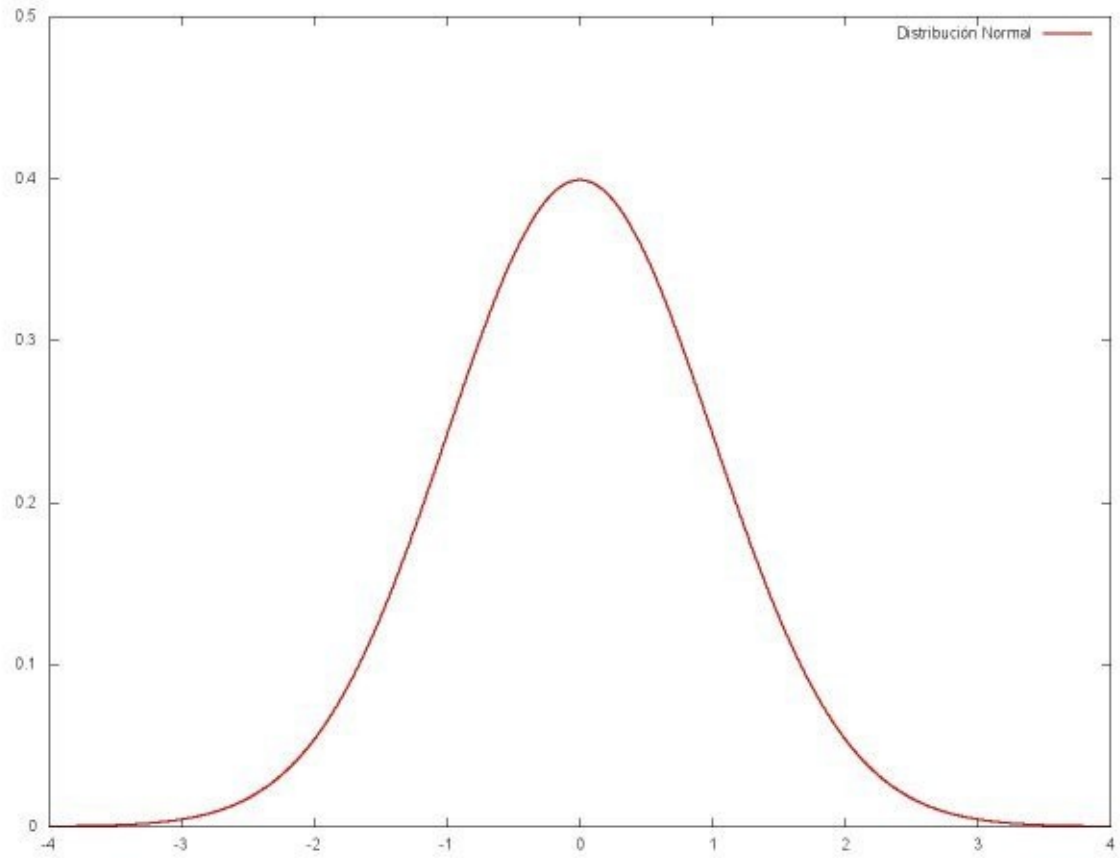
CALCULANDO UN ÚNICO VALOR

Como nuestra estimación debe ser finalmente un valor debemos calcularlo a partir de los tres valores anteriores.

Para ello deberemos aproximarlos a través de una distribución de probabilidad que tenga en cuenta los casos determinados anteriormente.

La fórmula que podemos utilizar para ello, varía dependiendo de la distribución de probabilidad con la que queramos aproximar los valores. Veámoslo en cada caso:

Distribución Normal

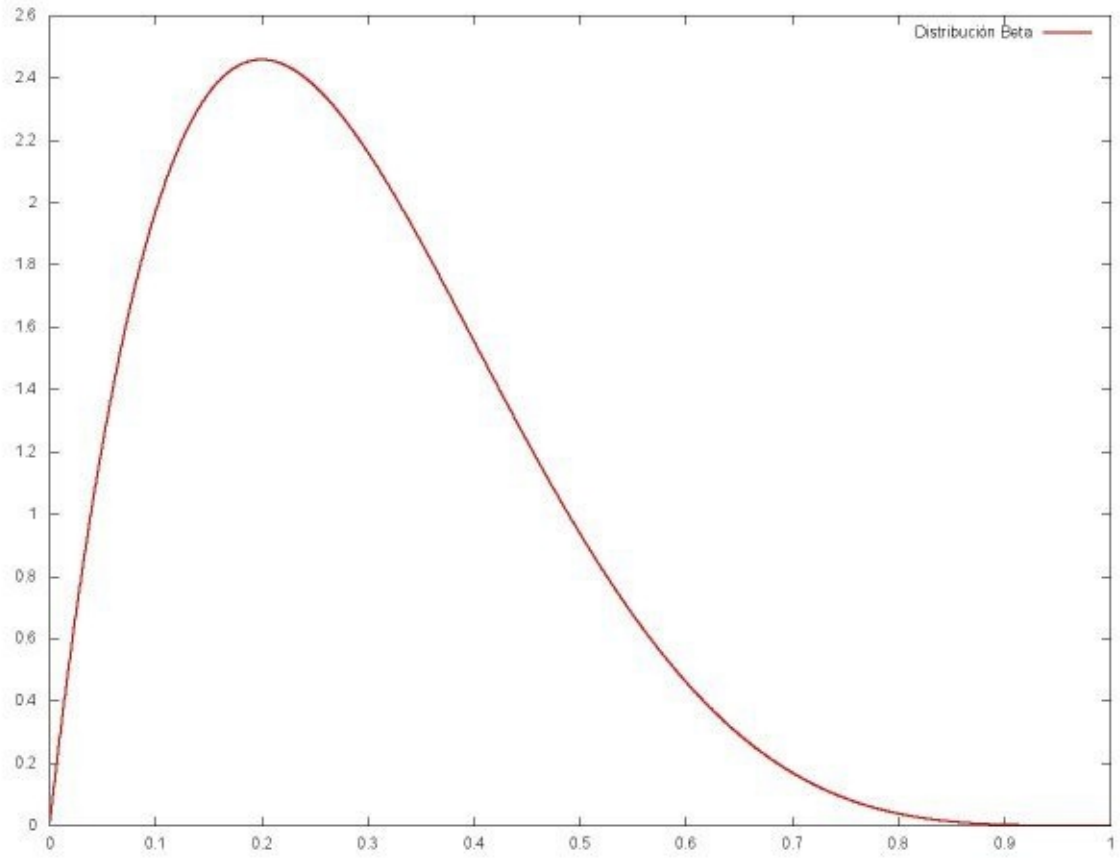


La distribución normal también se conoce por el nombre de campana de Gauss, debido a la forma especial que tiene similar a una campana. Es una distribución simétrica: el valor más probable está a la misma distancia del pesimista que del optimista. Si aproximamos con una distribución normal fórmula será:

$$E = O + \frac{(P - O)}{2}$$

$$DS = \frac{(P - O)}{6}$$

Distribución Beta (β)



Es una distribución asimétrica donde el valor más probable ya no está en el centro sino que está desplazado hacia uno de los extremos. Si aproximamos con una distribución beta (doble triangular) la fórmula será:

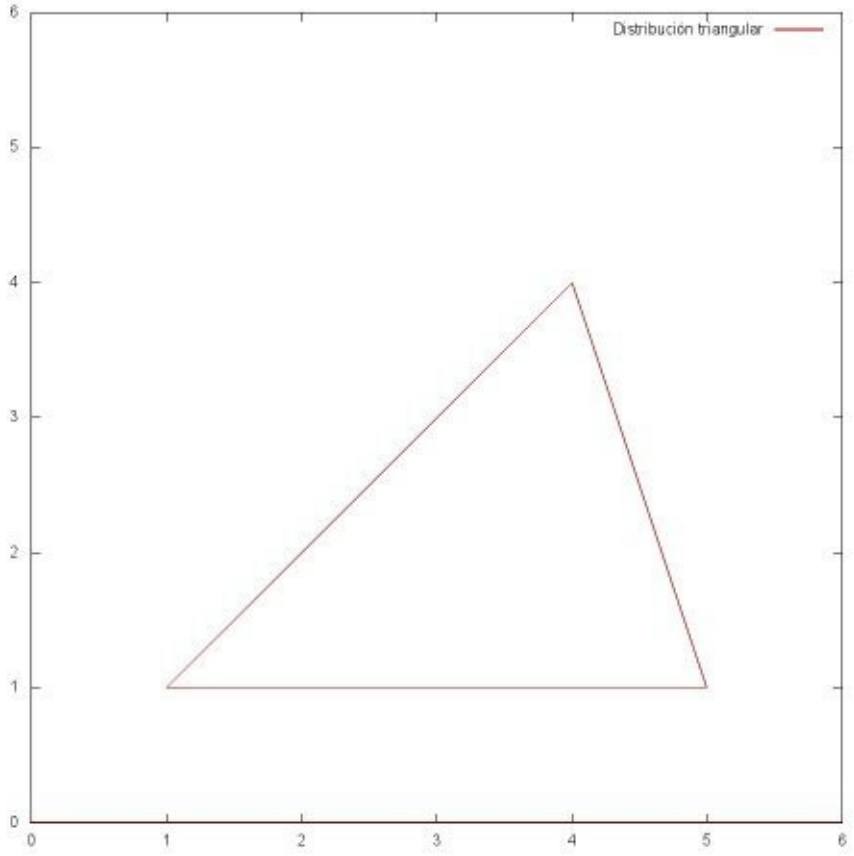
$$\boldsymbol{E} = \frac{(P + 4 \cdot \mathrm{M1} + O)}{6}$$

$$\boldsymbol{DS} = \frac{(O - P)}{6}$$

En este caso, se interpreta que el caso *Más Probable*, como su nombre indica, será el que tenga más posibilidades de acontecer por lo que se le da un peso más importante que a los otros dos.

Ésta es la fórmula que se utiliza habitualmente.

Distribución Triangular



La diferencia principal con respecto a la distribución beta es que a los 3 valores se les da el mismo peso, no se hace prevalecer el más probable. Si aproximamos con una distribución triangular la fórmula será:

$$\boldsymbol{E} = \frac{(P + \mathrm{Ml} + O)}{3}$$

$$\mathbf{DS} = \sqrt{\frac{(O - P)^2 + (\mathrm{Ml} - O) \cdot (\mathrm{Ml} - P)}{18}}$$

Aquí los tres valores tienen el mismo peso en la estimación final que se indique.

CONCLUSIÓN

Con los valores de la ***Estimación*** (**E**) obtenida y de la ***Desviación*** **estándar o típica** (**DS**) podemos informar una estimación y el rango de valores más probable donde se moverá la misma.

Como dato adicional, si fijáramos los mismos parámetros para las tres distribuciones descritas, los valores obtenidos serían cada vez más pesimistas en este orden:

- Beta
- Triangular
- Normal

Técnicas Delphi

Creada en los años cuarenta en EE.UU, la técnica Delphi se comienza a utilizar en el campo del software a partir de **1981** por *Barry Boehm*.

Los pasos de que consta son los siguientes:

- Un coordinador proporcionada a cada experto una especificación del proyecto propuesto y un documento para indicar su opinión.

- Los expertos completan el documento de forma anónima. Pueden hacer preguntas al coordinador pero no pueden hacerse preguntas entre ellos.

- El coordinador ofrece a cada experto el valor medio de las opiniones recogidas y se les pide una nueva estimación anónima indicando las razones de las posibles modificaciones.

- Se repite el proceso de recogida de opiniones hasta llegar a un consenso. No se realizan reuniones en grupo en ningún momento.

Un refinamiento de esta técnica denominado **Delphi de banda ancha** fue popularizado también por *Barry Boehm*es su libro *Software Engineering Economics*(1981) . Se denomina de banda ancha debido a que hay un mayor número de interacciones entre los expertos.

Los pasos a seguir son:

1) El coordinador proporcionada a cada experto una especificación del proyecto propuesto y un documento para indicar su opinión.

2) El coordinador reúne a los expertos para que intercambien puntos de vista sobre el proyecto.

3) Los expertos rellenan el documento de forma anónima.

4) El coordinador ofrece a cada experto el valor medio de las opiniones recogidas y se les pide una nueva estimación anónima, sin indicar las razones de las posibles modificaciones.

5) El coordinador convoca una reunión para que los expertos discutan las razones de las diferencias entre sus estimaciones.

6) Se rellenan anónimamente los impresos y se repite los puntos 4, 5 y 6 hasta llegar a un consenso.

La **ventaja** principal de esta técnica es que r**ecoge la opinión experta sobre el proyecto** actual basada en experiencias anteriores difíciles de evaluar por otros medios (características especiales del personal, peculiaridades del proyecto,...).

El principal **inconveniente** se encuentra en la subjetividad o inexperiencia de las personas elegidas en la consulta, es decir **la selección adecuada de los expertos**.

COCOMO 81

COCOMO (Cost Constructive Model) es el Modelo Constructivo de Costes. Un modelo de dominio público, que viene siendo utilizado y evaluado ampliamente.

ORIGEN

Fue desarrollado por *Barry Boehm* a finales de los años **70** comienzo de los **80** y descrito en su libro *Software Engineering Economics* (**1981**). Se conoce como COCOMO I ó COCOMO 81.

INTRODUCCIÓN

COCOMO 81 asume que el modelo de desarrollo que se utiliza es en Cascada y que se utilizan lenguajes imperativos como C, Pascal, etc.

Es un modelo matemático de base empírica que permite la estimación del coste y la duración de los Proyectos de Software: **esfuerzo y tiempo**.

Es empírico debido a que se basa en ecuaciones no lineales obtenidas mediante técnicas de regresión a través de un histórico de proyectos ya realizados (finalizados).

En el modelo de COCOMO se establecen tres tipos posibles de proyectos sobre los que podemos elegir el que más se ajuste a nuestra situación.

A estos tipos de proyecto se les denomina los **modos del modelo**:

- **Modo Orgánico.** Este caso se corresponde con proyectos sencillos, proyectos en los cuales se tiene mucha experiencia desarrollándolos y cuyo entorno es estable. La dimensión del proyecto suele ser de hasta 50.000 LDC (Líneas de Código).

- **Modo Semi-acoplado o semi-encajado.** La complejidad de los proyectos es superior al anterior. Además, el equipo está formado a partes iguales por personas con experiencia y personas sin ella.

Pueden llegar a tener una dimensión de 300.000 LDC.

- **Modo Empotrado.** El problema a resolver es distinto a los tratados en la experiencia del equipo, es un problema único.

Son los proyectos más complejos donde la experiencia del equipo es limitada sino nula. Pueden incluir grandes innovaciones técnicas.

En función de la precisión que necesitemos en la estimación, podemos utilizar tres desarrollos diferentes del modelo.

Para todos ellos las fórmulas de cálculo son las mismas:

$$E = a \cdot (KLOC)^b \cdot m(X)$$

$$Tdev = c \cdot (E)^d$$

$$P = \frac{E}{(Tdev)}$$

Donde:

- **E**. Es el esfuerzo medido en personas/mes
- **Tdev**. Es el tiempo estimado en meses
- **P**. Es el número de personas requerido para el proyecto
- **a, b, c, d**. Son constantes con valores definidos según cada modo y cada modelo
- **KLOC**. Es el número de miles de líneas de código fuente que tiene el

software que estamos intentando estimar

A continuación vamos a ver los tres desarrollos del modelo que podemos seleccionar en función de la precisión que necesitemos en los datos de nuestra estimación.

MODELO BÁSICO

Se basa en el tamaño expresado en miles de líneas de código (KLOC).

Las fórmulas del modelo son las siguientes:

$$E = a \cdot (KLOC)^b$$

$$Tdev = c \cdot (E)^d$$

$$P = \frac{E}{(Tdev)}$$

Donde:

- **E**. Es el esfuerzo medido en personas/mes
- **Tdev**. Es el tiempo estimado en meses
- **P**. Es el número de personas requerido para el proyecto
- **a, b, c, d**. Son constantes con valores definidos según cada modo y cada modelo
- **KLOC**. Es el número de miles de líneas de código fuente que tiene el

software que estamos intentando estimar

Los valores de las constantes son los siguientes:

MODO	a	b	c	d
Orgánico	2.40	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	3.60	1.20	2.50	0.32

Cuando queramos realizar una estimación, deberemos calcular el número de miles de líneas de código fuente de nuestro proyecto (KLOC) y utilizarlo como entrada en las fórmulas de más arriba, previa selección de los parámetros adecuados en función del tipo de proyecto en el que nos encontremos.

MODELO INTERMEDIO

Toma como entrada las miles de líneas de código (KLOC) y un multiplicador (m(X)). Este multiplicador se calcula a partir de 15 parámetros denominados **Atributos de Coste**.

Los *atributos de coste* nos permiten valorar el entorno de desarrollo del proyecto para tenerlo en cuenta en la estimación.

$$E = a \cdot (KLOC)^b \cdot m(X)$$

$$Tdev = c \cdot (E)^d$$

$$P = \frac{E}{(Tdev)}$$

Donde:

- **E**. Es el esfuerzo medido en personas/mes

- **Tdev**. Es el tiempo estimado en meses
- **P**. Es el número de personas requerido para el proyecto
- **a, b, c, d**. Son constantes con valores definidos según cada modo y cada modelo
- **m(X)**. Es un multiplicador que se calcula con los 15 atributos de coste que se indican más abajo
- **KLOC**. Es el número de miles de líneas de código fuente que tiene el software que estamos intentando estimar

Los valores de las constantes son los siguientes:

Modo	a	b	c	d
Orgánico	3.20	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	2.80	1.20	2.50	0.32

Los atributos de coste se asocian en 4 grupos: relativos al Software, relativos al Hardware, relativos a las Personas y relativos al Proyecto.

En total se definen 3 **atributos dependientes del software** que deben evaluarse en el modelo de COCOMO para poder realizar una estimación y son los siguientes:

- **RELY. Fiabilidad**

  Da una medida de las consecuencias que sufrirá el usuario si se produce un defecto en el funcionamiento del software. Es la garantía de funcionamiento que le exigimos al software.

- **DATA. Tamaño de la Base de datos**

  Es el valor relativo del tamaño de la Base de Datos con respecto al tamaño de la aplicación. Se obtiene dividiendo el tamaño en Bytes de la BD entre el tamaño en líneas de código del programa:

$$DATA = \frac{\text{Volumen BD}}{\text{Volumen Aplicación}}$$

- **CPLX. Complejidad**

Es la Complejidad del Producto.

Los rangos de los valores de los atributos son los siguientes:

Atributo	Valores					
	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
RELY	Un error leve	Pérdida facilmente recuperable por los usuarios	Pérdida moderada pero de la que se puede salir con facilidad	Gran pérdida financiera o inconvenien cia masiva humana	Implicaría la pérdida de vidas humanas	-
	0,75	0,88	1	1,15	1,4	-
DATA	-	0-10	10-100	100-1000	>1000	-
	-	0,94	1	1,08	1,16	-
CPLX	Se usan expresiones matemática s simples	-				Se usan muchos recursos de planificación
	0,7	0,85	1	1,15	1,3	1,65

En total se definen 4 **atributos dependientes del hardware** que deben evaluarse en el modelo de COCOMO para poder realizar una estimación y son los siguientes:

- **TIME. Restricciones de tiempo de ejecución**

Limitaciones en el % de uso de la CPU

- **STOR. Restricciones de memoria virtual**

Limitaciones en el % de uso de la memoria

- **VIRT. Volatilidad de la máquina virtual**

Cambios en el hardware y software de la máquina que se va a utilizar

- **TURN. Tiempo de respuesta**

Tiempo de respuesta requerido

Los rangos de los valores de los atributos son los siguientes:

Atributo	Valores					
	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
TIME	<5%	-	50%	-	-	>95%
	-	-	1	1,11	1,3	1,66
STOR	<5%	-	50%	-	-	>95%
	-	-	1	1,06	1,21	1,56
VIRT	-	0,87	1	1,15	1,3	-
TURN	-	Sistema Interactivo	-		Tiempo de respuesta > 12h	-
	-	0,87	1	1,07	1,15	-

En total se definen **5 atributos dependientes de las personas** que intervienen en el proyecto y que deben evaluarse en el modelo de COCOMO para poder realizar una estimación y son los siguientes:

- **ACAP. Capacidad de análisis**

Es la capacidad de los Analistas del proyecto para el análisis, eficiencia y cooperación.

- **AEXP. Experiencia en la aplicación**

Experiencia del equipo del proyecto en aplicaciones similares a la actual. Se toma el valor de la experiencia media del equipo

- **PCAP. Calidad de los programadores**

Valor de la capacidad de los Programadores

- **VEXP. Experiencia en la máquina virtual**

Dimensión de la experiencia del equipo en la Máquina virtual utilizada. Se toma el valor de la experiencia media del equipo

- **LEXP. Experiencia en el lenguaje**

Dimensión de la experiencia del equipo en el lenguaje de programación que se va a utilizar. Se toma el valor de la experiencia media del equipo

Los rangos de los valores de los atributos son los siguientes:

Atributo	Valores					
	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
ACAP	1,46	1,19	1	0,86	0,71	-
AEXP	<=4 meses	1 año	3 años	6 años	>= 12 años o re implement ación de un subsistem a	-
	1,29	1,13	1	0,91	0,82	-
PCAP	1,42	1,17	1	0,86	0,7	-
VEXP	<=1 mes	4 meses	1 año	>=3años	-	
	1,21	1,1	1	0,9	-	
LEXP	<=1 mes	4 meses	1 año	>=3años	-	
	1,14	1,07	1	0,95	-	

En total se definen **3 atributos dependientes del propio proyecto** que deben evaluarse en el modelo de COCOMO para poder realizar una estimación y son los siguientes:

- **MODP. Técnicas actualizadas de programación**

Utilización de prácticas modernas de programación. Aquí se entiende por prácticas modernas a la programación estructurada y al desarrollo top-

down, algo que hoy por hoy no tiene mucho sentido tratarlo de moderno

- **TOOL. Utilización de herramientas de software**

Utilización de herramientas de desarrollo de software

- **SCED. Restricciones de tiempo de desarrollo**

Restricciones en el cumplimiento de los plazos

Los rangos de los valores de los atributos son los siguientes:

Atributo	Valores					
	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
MODP	No se usan	Uso experimental de alguna de las técnicas	Experiencia razonable en alguna de las técnicas	Experiencia razonable en gran parte de las técnicas	Uso habitual de las técnicas	-
	1,24	1,1	1	0,91	0,82	-
TOOL	Se usan herramientas básicas	-			Se usan herramientas específicas	-
	1,24	1,1	1	0,91	0,83	-
SCED	Si vamos adelantados en cuanto a los plazos	-			Si vamos atrasados en cuanto a los plazos	-
	1,23	1,08	1	1,04	1,1	-

MODELO AVANZADO

Igual al anterior pero en lugar de mantener los valores de los 15 Atributos de Coste durante la duración del proyecto, éstos se recalculan para cada fase.

COCOMO 2000

La principal diferencia entre los dos versiones de COCOMO, es que COCOMO I supone que el modelo de desarrollo que se utiliza es en Cascada y utilizando lenguajes imperativos como C, Pascal, etc. y COCOMO II tiene en cuenta el modelo de desarrollo en espiral (prototipos) definiendo varios niveles que permiten obtener estimaciones detalladas de forma incremental.

ORIGEN

Su versión más reciente COCOMO II se publicó en el año 2000 y, por ello, se conoce como COCOMO II o COCOMO 2000. Editada por *Barry Boehm* y varios autores más y llamada *Software Cost Estimation with COCOMO II*.

NIVELES

Nivel de Construcción de Prototipos. Se supone que el sistema es creado a través de componentes reutilizables, scripts y programación en la BD.

Se utiliza para estimaciones de desarrollo de prototipos. El tamaño del software se mide en puntos de aplicación que es el mismo concepto que puntos objeto pero se le cambió el nombre para que no se confundiera con el desarrollo orientado a objetos.

$$PM = \frac{(NAP \cdot (1 - \frac{\%\text{Reutilización}}{100}))}{\text{PROD}}$$

Donde:

- **PM**: Esfuerzo estimado en personas/mes
- **NAP**: total de puntos aplicación en el sistema a desarrollar
- **%Reutilización**: es una estimación de la cantidad de código reutilizado en el desarrollo

- **PROD**: Productividad en puntos aplicación/mes

Nivel de Diseño Inicial. La estimación se realiza después de que se hayan establecido los requerimientos en fases tempranas. La medición se realiza en Puntos Función IFPUG. Se usa el estándar con 7 multiplicadores. La meta es realizar una estimación detallada sin demasiado esfuerzo. Es ideal para evaluar distintas alternativas en los requerimientos de usuario.

$$\text{Esfuerzo} = 2.94 \cdot \text{Tamaño}^{B} \cdot M$$

Donde:

- **Tamaño**. Medido en KSLOC (Nº de miles de líneas de código). Calculado a través de los Puntos Función.
- **B**. Refleja el esfuerzo creciente requerido para incrementar el tamaño del proyecto. Varía entre 1,1 a 1,24 dependiendo de la novedad del proyecto, la flexibilidad del desarrollo, los procesos para la resolución de riesgos, la cohesión del equipo de desarrollo y el nivel de madurez del proceso en la organización.
  El cálculo se ve en el nivel de post arquitectura.
- **M**. Se calcula con las 7 características. Se evalúan de 1 a 6 donde 1 es muy bajo y 6 es muy alto:
  - *RCPX*. Fiabilidad y complejidad del producto
  - *RUSE*. Reutilización requerida
  - *PDIF*. Dificultad de la plataforma
  - *PERS*. Capacidad del Personal
  - *PREX*. Experiencia del personal
  - *SCED*. Restricciones de plazos
  - *FCIL*. Facilidades de Apoyo

Nivel de Reutilización. Se calcula el esfuerzo necesario para integrar

componentes reutilizables y/o el código generado por herramientas de diseño. Suele utilizarse junto con el siguiente nivel.

Se tiene en cuenta dos tipos de reutilización de software:

- **Caja negra.** No hay que entender el software que se utiliza ni se tienen que realizar cambios en él.
- **Caja blanca.** Es el código que hay que adaptar para poder incluirlo en el desarrollo propio. Debemos entenderlo y modificarlo.

También se tiene en cuenta la generación de código automática desde el diseño con herramientas para ello:

$$\mathrm{PM(auto)} = \frac{\left(\mathrm{ASLOC} \cdot \frac{\mathrm{AT}}{100}\right)}{\mathrm{ATPROD}}$$

Donde:

- **PM(auto)**: Esfuerzo estimado para el código reutilizado en personas/mes.
- **AT**. % de código adaptado que se genera automáticamente.
- **ATPROD**. Productividad de los desarrolladores que integran el código. Bohem indica que es entorno a 2.400 líneas código / mes.
- **ASLOC**. Nº de líneas de código en los componentes que deben ser adaptados. No contiene el código generado automáticamente.

Cuando en la estimación tenemos que tener en cuenta código nuevo y código reutilizado lo que se suele hacer es calcular el valor equivalente en líneas de código nuevo para las reutilizadas, sumarlas al total y realizar la estimación con respecto a ese valor:

$$\mathrm{ESLOC} = \mathrm{ASLOC} \cdot \left(1 - \frac{\mathrm{AT}}{100}\right) \cdot \mathrm{AAM}$$

Donde:

- **ESLOC**. Nº Equivalente en líneas de código nuevo
- **ASLOC**. Nº de líneas de código en los componentes que deben ser adaptados. No contiene el código generado automáticamente
- **AAM**. Se conoce como Ajuste de la adaptación y contabilizado el esfuerzo de reutilización del código se calcula como la suma de 3 valores:
    - *AAF ó Componente de Adaptación*. Es el coste de realizar cambios en el código reutilizado: diseño, código e integración.
    - *SU ó componente de comprensión*. Es el coste de comprender el código a reutilizar y la familiaridad del desarrollador con el mismo. El rango es de 50 para código complejo no estructurado a 10 para código orientado a objeto bien escrito.
    - *AA ó factor de cálculo*. Es el coste de la toma de decisiones para la reutilización del código. El Rango es entre 0 y 8 según el esfuerzo requerido.

Nivel de Post arquitectura. Una vez realizado el diseño, la estimación resulta más precisa. Para ello se utiliza la fórmula estándar con 17 multiplicadores.

Atributos relativos al **PRODUCTO**.

Código	Atributo	Establecido en COCOMO 81
RELY	Fiabilidad	SI
CPLX	Complejidad	SI
DOCU	Amplitud de la documentación requerida	NO
DATA	Tamaño de Base de datos	SI
RUSE	% de componentes reutilizables requeridos	NO

Atributos relativos a la **PLATAFORMA**.

Código	Atributo	Establecido en COCOMO 81
TIME	Restricciones de tiempo de ejecución	SI
PVOL	Volatibilidad de la Plataforma de Desarrollo	NO
STOR	Restricciones de memoria virtual	SI

Atributos relativos a las **PERSONAS**.

Código	Atributo	Establecido en COCOMO 81
ACAP	Capacidad de análisis	SI
PCON	Continuidad del Personal	NO
PCAP	Calidad de los programadores	SI
PEXP	Experiencia de los programadores en el dominio de la aplicación	NO
AEXP	Experiencia en la aplicación	SI
LTEX	Experiencia en el lenguaje y en las herramientas de desarrollo	NO

Atributos relativos al **PROYECTO**.

Código	Atributo	Establecido en COCOMO 81
TOOL	Utilización de herramientas de software	SI
SCED	Restricciones de tiempo de desarrollo	SI
SITE	Ámbito de los distintos lugares de trabajo y sus comunicaciones	NO

Es el más detallado y se utiliza cuando conocemos el diseño de la arquitectura del sistema.

La fórmula sigue siendo la misma pero en lugar de 7 factores para precisar el multiplicador se utilizan 17, debido a que disponemos de más información que en las fases iniciales.

Cada estimación del número de líneas de código se calcula como la suma de 3 elementos:

- Estimación del número de líneas de código nuevas a desarrollar.

- Estimación del número de líneas de código equivalentes para el código que se va a reutilizar (**ESLOC**).
- Estimación del número de líneas de código que deben modificarse por cambios en los requisitos.

El tamaño total del código es la suma de las tres (**KSLOC**).

El valor **B** (el elevado) de la fórmula en COCOMO I era un rango de 3 posibles valores en función del modo en el que nos encontremos.

En COCOMO II es un valor continuo que se estima a partir de 5 factores de escala.

Cada factor se evalúa desde muy bajo (5) hasta extra alto (0) y se obtiene el valor con la siguiente fórmula:

$$B = 1.01 + \frac{\left(\sum_{i=1}^{5} \text{Factor}_i\right)}{100}$$

Donde los Factores son:

- **Precedentes.** Valora la experiencia previa de la organización en este tipo de proyectos. Muy bajo es sin experiencia previa. Extra alto significa que está completamente familiarizada con estos proyectos.
- **Flexibilidad de desarrollo.** Muy bajo es que se utiliza el proceso de desarrollo descrito. Extra alto significa que el cliente establece sólo metas generales.
- **Resolución de la arquitectura/riesgo.** Muy bajo, se hace poco análisis del riesgo. Extra alto se hace un análisis completo de los riesgos.
- **Cohesión del Equipo.** Indica cómo se conocen entre sí los miembros del equipo de desarrollo. Muy bajo significa interacciones difíciles. Extra alto significa un equipo integrado y efectivo sin problemas de comunicación.
- **Madurez del Proceso.** Equivalente al valor del nivel de madurez CMMi que alcance la organización.

Modelo de Putnam o SLIM

El modelo de Putnam es un modelo empírico de estimación del esfuerzo del software.

Originalmente fue descrito por Lawrence Putnam en un documento publicado en **1978**.

SLIM (*Software Lifecycle Management*) fue el nombre con el cual Putnam bautizó a una suite de herramientas propietarias desarrolladas basadas en este modelo.

El modelo se basa en el hecho de que Putnam detectó que los perfiles del personal de los proyectos de software seguían la distribución de Rayleigh y de ella extrajo la **ecuación del software**:

$$\frac{B^{\frac{1}{3}} \cdot \text{Tamaño}}{\text{Productividad}} = Esfuerzo^{\frac{1}{3}} \cdot Tiempo^{\frac{4}{3}}$$

Donde:

- **Tamaño**. Es el tamaño del producto. Putnam utilizada ESLOC (nº de líneas de código efectivas).
- **B**. Es un factor de escala que varía en función del tamaño del proyecto.
- **Productividad**. Es la capacidad de una determinada organización de software para producir software de un tamaño determinado con un ratio de errores concreto.
- **Esfuerzo**. Esfuerzo total del proyecto en personas/año.
- **Tiempo**. Tiempo total estimado para concluir el proyecto en años.

Para una determinada tarea la ecuación queda como:

$$\text{Esfuerzo} = \left[\frac{\text{Tamaño}}{\text{Productividad} \cdot \text{Tiempo}^{\frac{4}{3}}} \right]^3 \cdot B$$

Donde el valor de B se puede obtener de la tabla siguiente en función del tamaño en Líneas de Código Fuente:

Tamaño (SLOC en miles)	B
entre 5 y 15	0.16
20	0.18
30	0.28
40	0.34
50	0.37
>70	0.39

Planning Poker

Es una derivación de la técnica Delphi de Banda Ancha siendo utilizado comúnmente dentro del desarrollo ágil junto con el eXtreme Programming.

Fue descrito por primera vez por *James Grenning* en **2002** y difundido por *Michael Cohn* en su libro *Agile Estimating and Planning*.

Partimos de la necesidad de poder planificar una relación de **historias de usuario** y por ello debemos conocer la complejidad de cada una de ellas. Para asociar a cada historia de usuario una complejidad se utiliza una baraja de cartas.

La baraja de cartas está numerada siguiendo o bien la serie de Fibonacci o bien otras variaciones (ver el apartado dedicado a los Puntos de Historia). Por ejemplo, suelen encontrarse mazos enumerados como *0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100*.

El sentido de los valores definidos debe acordarse previamente. Pudiendo ser el mismo que en los puntos de historia: catalogar la complejidad relativa de las tareas con respecto a las demás, días de duración de la tarea, etc.

MÉTODO

Para realizar la estimación se reparte un mazo de cartas a cada uno de los estimadores y se mantiene una reunión para completar la estimación de las historias de usuario. Habrá un moderador que dirija la reunión pero no participe en las estimaciones (como vemos muy en línea con el método Delphy de banda ancha):

- **Presentación de la Historia**. Se selecciona una historia y el desarrollador que más la conozca hace una descripción de la misma. Se le pueden hacer preguntas y lanzar comentarios para aclarar cualquier circunstancia que se estime. Se registrará un resumen de lo discutido.
- **Estimación oculta**. Cada estimador coloca una carta boca abajo con su estimación. No deben discutirse los valores durante el debate.
- **Mostrar la estimación**. De forma simultánea todos los estimadores muestran su estimación.

- **Justificación de estimaciones altas y bajas**. Los estimadores que hayan emitido un valor más alto o más bajo que la media podrán explicar porque lo han hecho antes de comenzar de nuevo el debate.
- **Consenso**. Se repite el proceso realizando una nueva estimación oculta, hasta lograr el consenso de todos los estimadores. Particularmente se tiene en cuenta la opinión del desarrollador que realizará la tarea y la labor de negociación por parte del moderador.

Generalmente, se suele limitar el tiempo de cada ronda de póker para evitar enquistarse en un tema. Cuando se termina el tiempo se comienza de nuevo la ronda.

CONSIDERACIONES

Uno de los puntos positivos del Planning Poker es que cuanto mayor es el tamaño de la tarea, mayor diferencia hay entre los valores de la baraja y por tanto un mayor análisis debe ser realizado por parte del estimador.

Acelera las estimaciones limitando el número de opciones posibles.

Elimina un falso sentido de precisión en las estimaciones altas al tener un conjunto de valores dispar.

Si considera que una tarea debe tener un valor 25, dicho valor no existe en la baraja. En ella sólo tenemos 20 y 40 por tanto deberá analizar si realmente es lo suficientemente compleja como para marcarla con un 40 o si realmente con un 20 es suficiente.

Capítulo 3 Métodos de Medición en Puntos Función

Métodos de Medición en Puntos Función

Los Puntos Función son una herramienta que se utiliza en muchas empresas (y cada día en más) para la valoración de los proyectos. Cuando se habla en general de Puntos Función puede parecer que sólo hay un método y nada más lejos de la realidad.

La primera definición de los Puntos Función se la debemos a *Allan Albrecht*. En **1979**, trabajando para IBM, introdujo los conceptos que permitieron medir el software a través del conteo de la funcionalidad entregada al usuario y no del conteo de los aspectos técnicos del mismo. **El objetivo que perseguía era medir la funcionalidad entregada al usuario independientemente de la tecnología utilizada y de la fase del ciclo de vida del proyecto.**

El Método IFPUG–FPA es el método que ha continuado con el trabajo original de *Allan Albrecht*. A partir de entonces los métodos de medición en puntos función se han diversificado y se han intentado adaptar a distintas necesidades del mercado y a solucionar distintos problemas que se presentaban para los primeros métodos definidos. Por ejemplo, COSMIC-FFP se ha adaptado para medir con mayor precisión aplicaciones en tiempo real.

Aunque el número de variaciones sobre el método de puntos función es amplio vamos a tratar los más destacados.

IFPUG FPA

El Método de Análisis en Puntos Función (FPA, Function Point Analysis) de IFPUG es el directo heredero del método creado por *Allan Albrecht*. IFPUG es el International Function Points Users Groups o, en español, el Grupo Internacional de Usuarios de Puntos Función.

Su sitio web es el siguiente: http://www.ifpug.org

ORIGEN

El grupo fue creado en 1986 para mejorar el método previamente definido por *Albrecht*. El nombre oficial del método es *Function Point Analysis* o FPA (Análisis en Puntos Función).

IFPUG mantiene un Manual donde describe el método de conteo en Puntos Función es el denominado *Counting Practices Manual*. La versión actual es la 4.3.1 y el original en inglés está traducido al chino, español, francés, italiano, coreano y portugués. Siendo la última versión de 2010.

Como primera versión del manual se considera un documento formal creado por *Albrecht* para IBM en 1984 denominado *IBM CIS & Guideline 313, AD/M Productivity Measurement and Estimate Validation*, fechado el 1 de Noviembre de **1984**.

La segunda versión del manual es la primera realizada por IFPUG en Abril de **1988** y ya se denominó *Internacional Function Point Users Group Function Point Counting Practices Manual.*

La tercera versión apareció en abril de **1990** siendo una evolución importante en el desarrollo del método. Fue el primer paso en el camino de establecer un

verdadero estándar a lo largo de distintas organizaciones, dejando de un lado una colección de interpretaciones de las reglas y estableciendo un documento coherente en su desarrollo de la representación, consensuada por la industria, de las reglas del método.

La cuarta versión apareció en enero de **1994** que también significó un paso importante en la evolución del método sobre todo con la introducción de la estimación del tamaño del proyecto en etapas tempranas del desarrollo, utilizando información de disciplinas de ingeniería. En esta versión del manual, se incluyeron numerosos ejemplos de aplicación del método debido a que las situaciones en las cuales había que aplicarlo se dispararon: incremento del uso de aplicaciones con ventanas con Graphics User Interface (GUI).

En enero de **1999**, apareció la versión 4.1 del manual. En ella se abogó por una clarificación del método simplificando reglas anteriores o añadiendo nuevas reglas o consejos para situaciones que antes no se habían documentado. Todo ello con el fin de facilitar el entendimiento del método por un mayor número de usuarios. Esta versión, salvo la parte referida a las características generales del sistema, cumplía los estándares ISO y por tanto se convirtió en estándar.

En enero de **2004** aparece la versión 4.2 del manual que no modifica ninguna regla anterior pero añade aclaraciones y mejoras a las interpretaciones de las reglas. También se agrupan dentro del manual una serie de documentos paralelos que se habían ido desarrollando para dar respuesta a algunas de las cuestiones planteadas por la industria en cuando a FPA:

- **Counting Logical Files** (Septiembre de 2001)
- **FPA in an Enhancement Environnement** (Abril de 2002)
- **Counting Code Data** (Septiembre de 2003)
- **Counting Shared Data** (Septiembre de 2003)

Finalmente, en Enero de **2009** aparece la última versión del manual hasta la fecha, la 4.3. En ella, no se han modificado ninguna de las reglas existentes y se ha intentado aclarar aquellas reglas que presentan una interpretación más difícil. También se ha intentado hacer más consistentes los resultados que se obtengan entre distintos CFPS (Certified Function Points Specialist).

INTRODUCCIÓN

Este método se basa en dividir la funcionalidad del sistema a medir en dos tipos de funciones:

- **Funciones transaccionales.** Estas funciones también llamadas *procesos elementales* intentan modelar las necesidades de proceso del usuario, entendiendo por usuario a cualquier persona o cosa que interactúa con el sistema que estamos midiendo. Ejemplos de funciones transaccionales son*:*

alta de empleado
listado de empleados
informe mensual de empleados
etc

- **Funciones de Datos.** Estas funciones tratan de modelar las necesidades de almacenamiento de información que tiene el usuario (usuario definido como en el punto anterior). Las funciones de datos también se denominan *Grupos Lógicos* y son una generalización del concepto de Entidad pero debemos tener cuidado porque aunque pueden coincidir no siempre es así.

El **valor de una medición en puntos función** de un sistema es la suma del valor de los puntos función de sus funciones transaccionales y de sus funciones de datos.

El valor en puntos función de cada una de las funciones se determina a través de sus componentes.

Para las **funciones transaccionales** sus componentes son:

- los atributos que envía o recibe la propia función. Por ejemplo:

los campos de un formulario
el resultado de una consulta
etc

- los Grupos Lógicos o Funciones de datos que utiliza. Por ejemplo:

la entidad de Empleado al dar de alta un elemento nuevo
etc

Para las **funciones de datos** sus componentes son:

- los atributos que almacenan. Por ejemplo:

código del empleado
nombre del empleado
departamento del empleados
etc.

- los grupos de atributos que se pueden identificar siguiendo unas determinadas reglas definidas en el método. Por ejemplo:

Empleado
Beneficiarios del Empleado
etc.

Estos valores permiten establecer la complejidad de cada función según unas tablas prefijadas en el método. Los posibles valores de la complejidad son Baja, Media o Alta.

Según el tipo de Función transaccional o de datos la complejidad se convierte en un valor en puntos función a través de unas tablas definidas en el método.

Aplicando todo ello obtenemos una valoración de nuestro proyecto en Puntos Función que se denominan **No Ajustados**.

Hasta aquí sería el estándar ISO del método de IFPUG aunque el método incluye una serie de pasos más que están cayendo en desuso y es el conocido como factor de ajuste.

El factor de ajuste es un valor que se calcula para cada sistema/aplicación que estemos midiendo, evaluando para ello 14 características del mismo.

Los valores de cada característica van de 0 a 5. Se suman todos y se les aplica una fórmula que puede hacer variar la valoración de Puntos Función No Ajustados en un -35% o +35%.

Las 14 características generales del sistema son:

1. **Comunicaciones de Datos**
2. **Procesamiento de Datos Distribuido**
3. **Rendimiento**
4. **Configuración Altamente Utilizada**
5. **Tasa de Transacciones**
6. **Entrada de Datos On-line**
7. **Eficiencia del Usuario Final**
8. **Actualización On-line**
9. **Complejidad de Procesamiento**
10. **Reusabilidad**
11. **Facilidad de Instalación**
12. **Facilidad de Operación**
13. **Múltiples Localizaciones**
14. **Facilidad de Cambio**

Con ello tendríamos los Puntos de Función Ajustados para nuestro proyecto.

CONCEPTOS

Antes de pasar a ver el método de conteo en Puntos Función de IFPUG en detalle, debemos introducir una serie de conceptos que nos permitirán entender mejor el método. De hecho, el propio método se basa en la definición y acotación de dichos conceptos para crear un método consistente y repetible, para que cualquier persona que lo estudie pueda obtener los mismos resultados en su aplicación.

Una pieza vital del método de IFPUG es el usuario. Toda la identificación de funciones de datos y funciones transaccionales se basa en esta figura y en la percepción que tiene de la aplicación o sistema.

Un ***usuario*** es cualquier persona que especifica los Requerimientos Funcionales de Usuario y/o cualquier persona o cosa que se comunique o interactúe con la aplicación en cualquier momento. Un usuario puede ser tanto un operador que utiliza las pantallas de la aplicación como otro sistema que se comunica a través de interfaces de datos.

Tan importante como el usuario es el concepto de ***punto de vista del usuario*** que se entiende como una descripción formal de las necesidades de negocio del usuario, hecha en su propio lenguaje. Esta descripción será la utilizada por los desarrolladores para proveer una solución en un lenguaje técnico y una vez acordada entre usuarios y desarrolladores será la que se utilice para realizar el conteo en puntos función. La forma física del punto de vista del usuario puede variar, desde propuestas a documentos de requisitos pasando por especificaciones detalladas, manuales de usuario, etc.

Otro concepto importante es el de ***identificable por el usuario***. Se refiere a los requisitos acordados y entendidos por desarrolladores y por usuarios, definidos para los procesos y/o los grupos de datos del sistema.

Hasta este punto, sabemos que es lo que determina y define cada elemento dentro del software, ahora es necesario que centremos y acotemos el objeto del conteo. Para ello, primeramente debemos tener claro cuál es el ***propósito de la medición***. El propósito nos determinará el tipo de medición de puntos función que deberemos realizar, así como el alcance de la medición para obtener la solución al problema de negocio tratado.

Teniendo claro el propósito de una medición, podemos pasar a fijar el ***alcance de la medición***. Éste nos determina que funcionalidad es la que está dentro del

estudio que estamos realizando. Define el conjunto o subconjunto del software que está siendo medido. Podría incluir a más de una aplicación.

Según su propósito el *alcance de la medición* puede ser o sólo las funciones utilizadas por el usuario o todas las funciones entregadas.

Para poder medir la aplicación, además del propósito y del alcance, debemos definir los ***límites de la aplicación***. Debemos saber dónde empieza y dónde termina para conocer que es interno a ella y que es externo. Los límites de la aplicación son una frontera entre el software que se está midiendo y el usuario. Son independientes de las consideraciones técnicas o de implementación.

LAS FUNCIONES

Establecido el donde, debemos fijarnos ahora en qué tenemos que contar. El objeto de medición son las funciones de datos y las funciones transaccionales de la aplicación.

Las ***funciones de datos*** representan la funcionalidad proporcionada al usuario para satisfacer los requisitos de datos internos y externos. Entendiendo por ***fichero*** a un grupo de datos lógicamente relacionados y no a una implementación física de estos grupos de datos, tenemos dos tipos diferentes de funciones de datos:

- Los **Ficheros Lógicos Internos** (ILF de *Internal Logical File*). Grupos de datos ó información de control relacionados lógicamente, identificables por el usuario y mantenidos dentro de la aplicación. El propósito principal es almacenar datos mantenidos por uno ó más procesos elementales de la propia aplicación.

- Los **Ficheros de Interfaces Externos** (EIF de External Interface File). Grupos de datos o información de control lógicamente relacionados, identificables por el usuario pero mantenidos dentro de los límites de otra aplicación. El propósito principal de un EIF es almacenar datos referenciados por uno o más procesos elementales

dentro de los límites de la aplicación. De aquí que un EIF para nuestra aplicación debe ser un ILF de otra.

Que un grupo de datos sea **mantenido** por la aplicación viene a significar que la aplicación tiene la capacidad de modificar información de dicho grupo de datos a través de uno o varios procesos elementales. Modificar incluye creación, borrado, cualquier cambio en los datos, etc.

Las funciones de datos pueden contener datos e información de control. Se entiende por ***información de control*** aquellos datos que influyen sobre el comportamiento de un proceso elemental que se está midiendo, determinando aspectos como el qué, el cuándo o el cómo van a ser procesados los datos.

Las ***funciones transaccionales*** representan la funcionalidad proporcionada al usuario para el procesamiento de los datos a través de la aplicación.

Las funciones transaccionales se identifican como procesos elementales, siendo un ***proceso elemental*** la unidad de actividad más pequeña que tiene sentido para el usuario. Además debe ser autosuficiente y dejar a la aplicación en un estado consistente desde un punto de vista del negocio de la aplicación.

A través del ***propósito principal*** de la función transaccional, éstas se dividen en tres tipos. Bien sea este propósito mantener un ILF o alterar el comportamiento del sistema, bien sea mostrar información al usuario:

- **Entrada Externa** (EI de *External Input*). Es un proceso elemental que procesa datos o información de control que provienen de fuera de los límites de la aplicación cuyo propósito principal es mantener uno o más ILF y/o modificar el comportamiento del sistema.

- **Salida Externa** (EO de *External Output*). Es un proceso

elemental que envía datos o información de control fuera de los límites de la aplicación y cuyo propósito principal es presentar información al usuario conteniendo la lógica de proceso al menos una fórmula matemática o un cálculo, la creación de un dato derivado, el mantenimiento de al menos un ILF del sistema o alterar el comportamiento del mismo.

- **Consulta Externa** (EQ de *External Inquiry*). Es un proceso elemental que envía datos o información de control fuera de los límites de la aplicación y cuyo propósito principal es presentar información al usuario sin que su lógica de proceso contenta fórmulas matemáticas o cálculos, ni cree datos derivados, ni mantenga ningún ILF y no altere el comportamiento del sistema.

LÓGICA DE PROCESO

Cada proceso elemental realiza una serie de acciones solicitadas por el usuario para cumplir una tarea determinada, esto es la **lógica de proceso**.

Dentro de esta lógica de proceso se puede realizar cualquiera de las siguientes acciones y realizarlas en distintas alternativas y/u ocurrencias. Es decir, cada acción puede aparecer en un número diferente de repeticiones y aparecer según distintas condiciones:

1. **Realización de validaciones.**

Cuando se añade una determinada información y se quiere comprobar que lo datos son correctos antes de incluirlos en el sistema.

2. **Realización de fórmulas matemáticas y cálculos.**

Como el caso de totalizadores en informes.

3. **Conversión de valores equivalentes.**

Por ejemplo, en el caso de conversión de monedas obteniendo los valores

correspondientes de tablas sin realizar el cálculo.

4. **Filtrado y selección de datos utilizando los criterios indicados para comparar múltiples conjuntos de datos.**

Como en casos donde de una lista se seleccionan las filas que cumplen un determinado criterio.

5. **Análisis de las condiciones para determinar cuáles son aplicables.**

Inclusión de disyunciones en el procesamiento para determinar que acciones aplican o cuáles no.

6. **Actualización de uno ó más ILF.**

En el alta de cualquier entidad se actualiza el ILF correspondiente.

7. **Se referencian uno ó más ILF ó EIF.**

Cuando se obtiene un dato de un ILF ó EIF necesario para calcular otro valor.

8. **Recuperación de datos o información de control.**

Para mostrar cualquier atributo de una entidad se accede a la misma para obtener sus datos.

9. **Creación de datos derivados mediante la transformación de los datos existentes para crear información adicional.**

Por ejemplo, la generación de un nombre de usuario con las 3 primeras letras de los dos apellidos de un empleado.

10. **Alteración del comportamiento del sistema.**

Por ejemplo, si un proceso se ejecuta mensualmente y realizamos una modificación para que se ejecute trimestralmente.

11. Preparación y presentación de información fuera de los límites de la aplicación.

Por ejemplo, una lista cualquier mostrada al usuario.

12. Capacidad para aceptar datos o información de control que atraviesen los límites de la aplicación.

Por ejemplo, los datos incluidos para el alta de una entidad.

13. Reordenación o reubicación de los datos.

Se solicita una lista ordenada por uno de sus campos y que los campos aparezcan en una determinada posición.

Dependiendo del tipo de función transaccional, alguna de las acciones de la lógica de proceso será obligatoria, opcional o no estará permitida. A continuación mostramos una tabla con el resumen de las mismas (ver Tabla):

Lógica de Proceso /Tipo de Función Transaccional		Tipo de Función		
		EI	EO	EQ
Acción de lógica de proceso	1. Realización de validaciones	x	x	x
	2. Realización de fórmulas matemáticas y cálculos	x	o*	-
	3. Conversión de valores equivalentes.	x	x	x
	4. Filtrado y selección de datos utilizando los criterios indicados para comparar múltiples conjuntos de datos	x	x	x
	5. Análisis de las condiciones para determinar cuales son aplicables	x	x	x
	6. Actualización de uno ó más ILFs	o*	o*	-
	7. Se referencian uno ó más ILFs ó EIFs	x	x	ob
	8. Recuperación de datos o información de control	x	x	ob
	9. Creación de datos derivados mediante la transformación de los datos existentes para crear información adicional	x	o*	-
	10. Alteración del comportamiento del sistema	o*	o*	-
	11. Preparación y presentación de información fuera de los límites de la aplicación	x	ob	ob
	12. Capacidad para aceptar datos o información de control que atraviesen los límites de la aplicación	ob	x	x
	13. Reordenación o reubicación de los datos	x	x	x

Donde:

- **x**. La acción es opcional.
- **o*** Al menos una de las acciones señaladas así, es obligatoria.
- **ob** La acción es obligatoria.
- **-** La acción está prohibida.

LOS DET, LOS RET Y LOS FTR

Como hemos visto hasta ahora, nuestro sistema se va a componer de una serie de ficheros lógicos y procesos elementales en función del propósito de nuestra medición, con respecto al alcance pretendido y determinados por los límites establecidos.

Ahora necesitamos medir dichos ficheros lógicos y procesos elementales para poder conferir a nuestro sistema un valor funcional determinado.

Necesitamos introducir varios conceptos antes de continuar: los DET, los RET y los FTR.

Un ***DET*** (*Data Element Type*) o ***Elemento del Tipo Dato*** es un campo único, no repetido, reconocible por el usuario.

Un ***RET*** (*Register Element Type*) o ***Elemento del Tipo Registro*** es un subgrupo de elementos de datos reconocible por el usuario, dentro de un ILF o un EIF.

Dentro de los RET podemos distinguir dos tipos:

- **RETs opcionales** o subgrupos opcionales son aquellos en los cuales el usuario puede elegirlo o no al crear una ocurrencia de los datos.

Por ejemplo:

Al dar de alta un empleado podemos añadir el nombre de sus hijos y edades.

Si no tuviera hijos no los añadiríamos

- **RETs obligatorios** o subgrupos obligatorios son aquellos en los cuales el usuario debe utilizar al menos uno.
  Por ejemplo:

Al dar de alta un empleado tenemos que indicar si es personal propio o subcontratado

Un ***FTR*** (*File Type Reference*) o ***Fichero Referenciado*** es un fichero lógico

interno (ILF) leído o mantenido por una función transaccional o es un fichero de interfaz externo (EIF) leído por una función transaccional.

Las funciones de datos se miden en Puntos Función de acuerdo a los DET y RET que se les identifiquen.

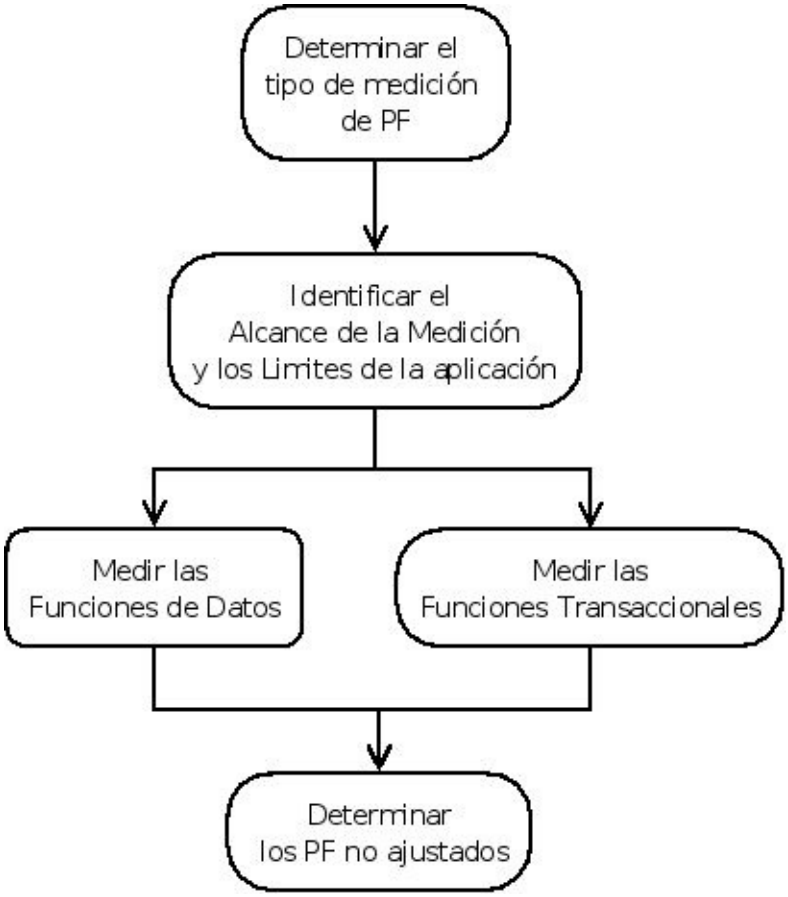
Las funciones transaccionales se miden en Puntos función de acuerdo a los DET y FTR que se les identifiquen.

PUNTOS FUNCIÓN NO AJUSTADOS

El método de puntos función se basa en el conteo de los conceptos que hemos desarrollado anteriormente, siguiendo una serie de pasos.

El método completo consta de 7 pasos y el método aprobado como estándar internacional consta de 5 pasos. La parte que no se incluye es la más técnica y alejada de la funcionalidad, correspondiente al VAF y los Puntos de Función Ajustados.

Vamos a ver primeramente el método estándar ISO y posteriormente pasaremos a ver los dos pasos adicionales que componen el método completo para la obtención de los puntos de función ajustados (ver Ilustración).



Paso 1. Determinar el tipo de medición de Puntos Función

Las mediciones de puntos función pueden estar asociadas tanto a proyectos como a aplicaciones.

Existen tres tipos de medición:

- **Medición de puntos función de un proyecto de desarrollo.**

  En este tipo de medición de puntos función se miden las funciones proporcionadas al usuario por el proyecto en la primera instalación del software que se ha desarrollado.

- **Medición de puntos función de un proyecto de mejora.**

  En este tipo de medición de puntos función se miden las modificaciones a

la aplicación existente entregadas a la finalización del proyecto. Como modificaciones se entienden las acciones que incorporan, cambian o eliminan funciones de usuario.

- **Medición de puntos función de una aplicación**

  En este tipo de medición de puntos función se miden las funciones actuales que la aplicación proporciona al usuario. Este tipo de medición se refiere siempre a una aplicación existente y también se denomina medida de puntos función de *línea base* o *instalada.*

  Se puede conseguir directamente a través de la medición de la aplicación o partiendo de la medición de un proyecto de desarrollo como primera medida, para posteriormente actualizarla con las mediciones de todos los proyectos de mejora que hayan tenido lugar después.

Paso 2. Identificar alcance de la medición y límites de la aplicación.

Estableciendo el alcance de la medición definimos el conjunto /subconjunto del software que estamos midiendo. Con él determinamos las funciones que están incluidas en la medición porque dan respuesta al propósito de la medición.

A la hora de establecer los límites de la aplicación, debemos tener en cuenta que para determinarlo debemos basarnos en la visión del usuario. Los límites entre aplicaciones no se deben corresponder con separaciones técnicas o tecnológicas sino con interpretaciones de negocio, de cómo las ve el usuario.

El alcance de una medición no tiene influencia sobre los límites de la aplicación establecidos. Ambos son independientes entre sí.

Dentro del alcance de una medición pueden encontrarse más de una aplicación y por tanto deberán identificarse los límites de cada una de ellas.

Secuencialmente tendríamos que establecer:

- propósito de la medición
- alcance de la misma
- identificar los límites de la/s aplicación/aplicaciones.

Paso 3. Medir las funciones de datos

Este paso y el siguiente se pueden realizar en paralelo o en cualquier orden indistintamente, aunque es preferible comenzar por éste ya que la identificación de los ILF/ EIF influye en la complejidad de los procesos elementales, aunque no en su número.

Para encontrar los ILF del sistema tenemos que cumplir las siguientes reglas:

- El grupo de datos o información de control que identifiquemos es un grupo lógico **identificable por el usuario**.

- Además el grupo de datos es **mantenido por un proceso elemental dentro de los límites** de la aplicación.

Para los EIF del sistema, debemos validar que los grupos lógicos encontrados cumplen las siguientes reglas:

- El grupo de datos o información de control es un grupo lógico

identificable por el usuario.

- El grupo de datos es referenciado por la aplicación y es externo a ella.
- El grupo de datos no está mantenido por la aplicación
- El grupo de datos es un ILF de otra aplicación (está siendo mantenido por otra aplicación).

Una vez que tengamos identificados los ILF y EIF de la aplicación, debemos mesurar su tamaño funcional contando sus DET y RET, para ello debemos seguir una serie de reglas:

- **Reglas para el conteo de DET.**
    - Contar un DET por cada campo único, no repetido, reconocible por el usuario, mantenido o recuperado por un proceso elemental de un ILF o EIF. Por ejemplo:

Nombre de empleado (se muestra en la aplicación y además al dar de alta un empleado se actualiza)

    - Contar un DET por cada parte de información requerida por el usuario para establecer una relación con otro ILF o EIF. Por ejemplo:

Claves foráneas

    - Si dos aplicaciones mantienen y/o referencian el mismo ILF/EIF, aunque accediendo a un subconjunto diferente de DET, a la hora de medir el ILF/EIF para cada una de las aplicaciones sólo debemos tener en cuenta los DET de los que esa aplicación esté haciendo uso.

Es decir, los DET que el Grupo Lógico contenga pero que son

utilizados únicamente por otras aplicaciones, no deben contabilizarse dentro de ella.
Por ejemplo:

*La aplicación A referencia el Grupo Lógico **Cliente** para leer: Nombre del Cliente e Identificador de Cliente*

*La aplicación B referencia el Grupo Lógico **Cliente** para leer: Nombre del Cliente, Identificador de Cliente y Nº de Hijos.*

*La Aplicación A cuenta el ILF/EIF **Cliente** con 2 DET*
*La Aplicación B cuenta el ILF/EIF **Cliente** con 3 DET*

- **Reglas para el conteo de RET**
    - Contar un RET por cada subgrupo opcional u obligatorio del ILF o EIF que se esté contando.
    - Si no existe ningún subgrupo contar que el ILF/EIF tiene 1 subgrupo.

Una vez que tengamos establecidos los DET y RET de cada uno de los ILF y EIF debemos traducir dichos valores en un valor de complejidad.

Existen tres valores de complejidad *Baja, Media y Alta*.

Se calculan en función de la tabla siguiente (ver Tabla):

Complejidad ILFs y EIFs		DETs		
		de 1 a 19	de 20 a 50	51 ó más
RETs	1	Baja	Baja	Media
	2 ó 5	Baja	Media	Alta
	6 ó más	Media	Alta	Alta

Tabla - Cálculo complejidad para ILF y EIF en el método de IFPUG

Cada una de estas complejidades tiene asociado un valor en puntos función fijo que se sumará al total de la cuenta.

La complejidad en PF se calcula de forma diferente para los ILF y los EIF según la siguiente tabla (ver Tabla):

Pesos por complejidad y tipo		Complejidad		
		Baja	Media	Alta
Tipo Función Datos	ILF	7	10	15
	EIF	5	7	10

Tabla - Valor de PF por Complejidad de Función de Datos en el método de IFPUG

Una vez calculados los valores de las contribuciones en Puntos Función de cada una de las Funciones de Datos, los sumamos todos para obtener el total de puntos función de estos componentes.

Paso 4. Medir las funciones transaccionales

Lo primero que tenemos que hacer para medir las funciones transaccionales es identificar los procesos elementales. Una vez identificados, los catalogaremos según la tipología de cada uno de ellos. La diferencia entre tipologías viene determinada por el diferente propósito principal de cada uno de los procesos

elementales.

Recordemos que un proceso elemental debe:

- ser la menor unidad de actividad con sentido para el usuario
- debe ser autosuficiente y dejar a la aplicación en un estado consistente.

ENTRADA EXTERNA (EI)

Cuando el propósito principal de un proceso elemental es mantener un ILF o alterar el comportamiento del sistema estaríamos hablando de una **Entrada Externa (EI, External Input)**.

Además de que sea proceso elemental y que tenga como propósito principal el mencionado, un EI debe cumplir con las siguientes reglas:

- Los datos o la información de control se reciben desde fuera de los límites de la aplicación.

- Si los datos recibidos por la aplicación no contienen información que altere el comportamiento del sistema, debe al menos mantener un ILF (mantener en el sentido de alta, baja, modificación).

- Y debe cumplir al menos una de las siguientes reglas (para ser único):
    - La lógica de proceso es única con respecto al resto de Entradas Externas (EIs) de la aplicación.

    - El conjunto de datos elementales (DETs) identificado es diferente de los conjuntos identificados para el resto de entradas externas (EIs) de la aplicación.

    - Los ILF o EIF referenciados son diferentes de los referenciados por el resto de entradas externas (EIs) de la

aplicación.

DIFERENCIA ENTRE SALIDA EXTERNA Y CONSULTA EXTERNA

Cuando el propósito principal de un proceso elemental es presentar información al usuario estamos hablando de una Salida Externa (EO, External Output) o de una Consulta Externa (EQ, External inQuiry).

En este caso, la diferencia entre ambas es relativa a la lógica de proceso.

La EQ se encarga únicamente de recuperar la información y mostrarla, no puede hacer nada más con ella (salvo ordenarla). En cualquier otro caso, donde además de mostrar información se realice un cálculo, mantenga un ILF, etc. se entiende que es una Salida Externa (EO).

SALIDAS EXTERNAS (EO)

Las reglas que debe cumplir un proceso elemental para ser identificado como una **Salida Externa (EO, External Output)** son las siguientes:

- Envía datos o información de control fuera de los límites de la aplicación.

- Debe cumplir al menos una de las siguientes reglas (para ser EO):
    - La lógica de proceso del proceso elemental contiene al menos un cálculo o una fórmula matemática.
    - La lógica de proceso crea datos derivados.
    - La lógica de proceso mantiene al menos un ILF

- La lógica de proceso altera el comportamiento del sistema

- Y <u>debe cumplir al menos</u> una de las siguientes reglas (para ser único):

    - La lógica de proceso es única con respecto al resto de Salidas Externas (EOs) de la aplicación.
    - El conjunto de datos elementales (DETs) identificado es diferente de los conjuntos identificados para el resto de Salidas Externas (EOs) de la aplicación.
    - Los ILF o EIF referenciados son diferentes de los referenciados por el resto de Salidas Externas (EOs) de la aplicación.

CONSULTAS EXTERNAS (EQ)

Las reglas que debe cumplir un proceso elemental para ser identificado como una **Consulta Externa (EQ, External inQuiry)** son las siguientes:

- Envía datos o información de control fuera de los límites de la aplicación.

- <u>Debe cumplir todas</u> las reglas siguientes (para ser EQ):
    - La lógica de proceso del proceso elemental recupera datos o información de control de un ILF o un EIF.
    - La lógica de proceso <u>*no contiene*</u> ningún cálculo ni fórmula matemática.
    - La lógica de proceso <u>*no crea* datos</u> derivados.
    - La lógica de proceso <u>*no mantiene*</u> un ILF.
    - La lógica de proceso <u>*no altera*</u> el comportamiento del sistema.

- Y <u>debe cumplir al menos</u> una de las siguientes reglas (para ser único):

- La lógica de proceso es única con respecto al resto de Consultas Externas (EQs) de la aplicación.
- El conjunto de datos elementales (DETs) identificado es diferente de los conjuntos identificados para el resto de Consultas Externas (EQs) de la aplicación.
- Los ILF o EIF referenciados son diferentes de los referenciados por el resto de Consultas Externas (EQs) de la aplicación.

DETS Y FTR

Una vez detectadas las funciones transaccionales, debemos identificar los DET (Data Element Type) y los FTR (File Types Referenced) para poder determinar el tamaño funcional de los procesos elementales.

Para un **EI** identificaremos los DET siguiendo estas reglas:

- Contaremos un DET por cada campo no repetido, reconocible por el usuario que entra o sale de los límites de la aplicación y es requerido para completar la entrada externa (EI).

- Contaremos un DET por la capacidad de enviar un mensaje de error durante el proceso fuera de los límites del sistema, confirmar que el proceso finalizó o verificar que el proceso debería continuar.

- Contaremos un DET por la capacidad de especificar la acción a realizar, aunque haya más de una forma de hacerlo.

- No se contarán campos que son recuperados o derivados por el sistema y almacenados en un ILF durante un proceso elemental si los campos no cruzan los límites de la aplicación.

Para un **EI** identificaremos los FTR siguiendo estas reglas:

- Contaremos un FTR por cada ILF mantenido ó mantenido y leído.
- Contaremos un FTR por cada ILF o EIF leído

Para un **EO/EQ** identificaremos los DET siguiendo estas reglas:

- Contaremos un DET por cada campo no repetido, reconocible por el usuario que entra en la aplicación y es requerido para especificar cuándo, qué y/o cómo los datos son recuperados o generados por el proceso elemental.

- Contaremos un DET por cada campo no repetido, reconocible por el usuario que sale de los límites de la aplicación.

- Si un DET entra y sale de los límites de la aplicación lo contaremos una sola vez.

- Contaremos un DET por la capacidad de enviar un mensaje de error durante el proceso fuera de los límites del sistema, confirmar que el proceso finalizó o verificar que el proceso debería continuar.

- Contaremos un DET por la capacidad de especificar la acción a realizar, aunque haya más de una forma de hacerlo.

- No se contarán campos que son recuperados o derivados por el sistema y almacenados en un ILF durante un proceso elemental si los campos no cruzan los límites de la aplicación.

- No se contarán literales como DET.

- No se contarán variables de paginación o marcas generadas por el sistema.

Una vez que tenemos todos los procesos elementales catalogados e identificados sus DET y FTR, debemos obtener la complejidad siguiendo estas tablas (ver Tablas):

Complejidad Entrada (EI)		DETs		
		de 1 a 4	de 5 a 15	16 ó más
FTRs	0 ó 1	Baja	Baja	Media
	2	Baja	Media	Alta
	3 ó más	Media	Alta	Alta

Tabla - Complejidad EI en el método de IFPUG

Complejidad Salida (EO) y Consulta (EQ)		DETs		
		de 1 a 5	de 6 a 19	20 ó más
FTRs	0 ó 1	Baja	Baja	Media
	2 ó 3	Baja	Media	Alta
	4 ó más	Media	Alta	Alta

Tabla - Complejidad de EOs y EQs en el método de IFPUG

Y una vez determinada la complejidad obtenemos los puntos función en base a la tipología del proceso elemental y la complejidad detectada (ver Tabla).

Puntos Función por complejidad y tipo		Complejidad		
		Baja	Media	Alta
Tipo Función	Salida Externa (EO)	4	5	7
	Entrada Externa (EI) y Consulta Externa (EQ)	3	4	6

Tabla - Puntos Función por Complejidad y Tipo Función en el método de IFPUG

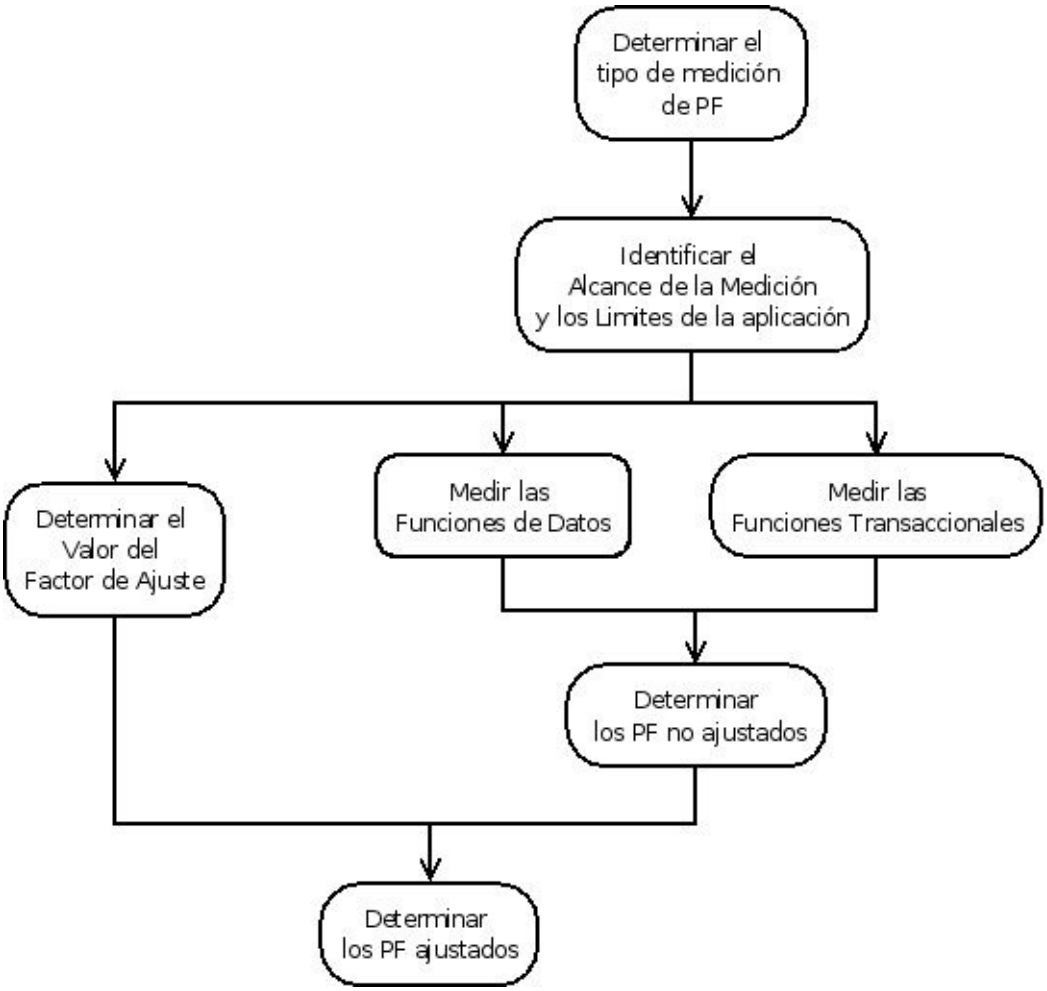
Paso 5. Determinar los Puntos de Función No Ajustados

Los puntos de función no ajustados reflejan la funcionalidad específica medible, proporcionada al usuario por el proyecto o la aplicación.

Una vez finalizadas las mediciones de las funciones de datos y de las funciones transaccionales tenemos los puntos de función no ajustados, como resultado de sumar el número de puntos función de cada una de las funciones medidas.

PUNTOS FUNCION AJUSTADOS

En este caso, partiríamos desde donde finalizó el método para el cálculo de los Puntos de Función No ajustados completando dos pasos más. El esquema global quedaría de la siguiente forma (ver Ilustración):



Paso 6. Determinar el valor del Factor de Ajuste

El **Valor del Factor de Ajuste** nos permite ponderar el valor de los Puntos de Función no ajustados en un +/- 35%.

Este valor se calcula evaluando el grado de influencia que tiene cada una de las

14 **Características Generales del Sistema ó GSCs** (*General System Caracteristics*) en el sistema que estamos midiendo.

El grado de influencia de cada característica va desde 0 – Sin influencia a 5 – Influencia Fuerte.

Una vez obtenidos todos los grados de influencia de las características del sistema, los sumamos para obtener el **Grado de Influencia Total ó TDI** (*Total Degree of Influence*).

$$\mathrm{TDI} = \sum_{i=1}^{14} GSC_i$$

Así, el Valor del Factor de Ajuste queda finalmente de la siguiente manera:

$$\mathrm{VAF} = (\mathrm{TDI} \cdot 0.01) + 0.65$$

Paso 7. Determinar los Puntos de Función Ajustados

Una vez calculado el Valor del Factor de Ajuste para la aplicación deberemos elegir la fórmula que mejor se ajuste al propósito de nuestro conteo.

Una vez tengamos la fórmula, la aplicaremos para realizar el cálculo de los puntos función ajustados.

DETERMINAR LA FÓRMULA DE MEDICION

Cada uno de los tipos de medición que podemos realizar en Puntos Función tiene una fórmula asociada.

De forma general, debemos tener en cuenta que toda medición estará formada por tres componentes:

- *Funcionalidad de la aplicación* descrita en los requisitos de usuario.

- *Funcionalidad de conversión* descrita en los requisitos de usuario.
- *Valor del factor de ajuste* de la aplicación.

Teniendo en cuenta esto, vamos a ver cada una de las fórmulas:

- **Fórmula para el cálculo de los Puntos Función de un Proyecto de Desarrollo**

La **funcionalidad de la aplicación** consiste en las funciones que se utilizarán después de la instalación del software para satisfacer las necesidades de negocio actuales del usuario.

La **funcionalidad de conversión** consiste en las funciones proporcionadas sólo durante la instalación para convertir datos y/o proporcionar otros requisitos de conversión especificados por el usuario, como pueden ser informes de conversión especiales.

El **valor del factor de ajuste** se calcula a partir de las 14 Características Generales del Sistema.

La fórmula para el cálculo de los puntos de función ajustados es la siguiente:

$$DFP = (UFP + CFP) \cdot VAF$$

Donde:

- **DFP** medición en puntos función del proyecto de desarrollo
- **UFP** medición de los puntos función no ajustados para las funciones disponibles después de la instalación.
- **CFP** medición de los puntos de función no ajustados añadidos con motivo de la conversión.
- **VAF** valor del factor de ajuste.

- **Fórmula para el cálculo de los Puntos Función de un Proyecto de Mejora**

La **funcionalidad de la aplicación** se compone de:

- Los puntos función identificados correspondientes a la funcionalidad añadida por las mejoras.
- Los puntos función medidos porque la funcionalidad existente se cambia durante el proyecto de mejora.
- Los puntos de función medidos que corresponden a la funcionalidad eliminada durante el proyecto de mejora.

La **funcionalidad de conversión** consiste en los puntos función proporcionados por cualquier requisito de conversión requerido por el usuario.

El valor del factor de ajuste no es uno sino que realmente son dos:

- El valor del factor de ajuste de la aplicación antes de que comience el proyecto de mejora.
- El valor del factor de ajuste de la aplicación después de que se complete el proyecto de mejora.

La fórmula para el cálculo de los puntos de función ajustados es la siguiente:

$$EFP = [(ADD + CHGA + CFP) \cdot VAFA] + (DEL \cdot VAFB)$$

Donde:

- **EFP** es la medición de puntos función del proyecto de mejora
- **ADD** es la medición de puntos función no ajustados de aquellas funciones añadidas por el proyecto de mejora.
- **CHGA** es la medición de puntos función no ajustados de aquellas funciones que fueron o serán modificadas por el proyecto de mejora. Este número refleja el tamaño de las funciones después de las modificaciones.
- **CFP** es la medición de puntos función de aquellas funciones

añadidas por la conversión.

- **VAFA** es el valor del factor de ajuste de la aplicación después del proyecto de mejora.
- **DEL** es la medición de puntos función no ajustados de aquellas funciones que fueron o serán eliminadas por el proyecto de mejora.
- **VAFB** es el valor del factor de ajuste de la aplicación antes de que el proyecto de mejora comience.

- **Fórmula para el cálculo de los Puntos Función de una Aplicación**

Existen dos variantes de esta fórmula. La primera de ellas establece la medición inicial de una aplicación y la segunda establece la medición de la aplicación después de un proyecto de mejora.

FÓRMULA PARA ESTABLECER LA MEDICIÓN INICIAL

En este caso, la medición de la aplicación no incluye requerimientos de conversión.

$$AFP = ADD \cdot VAF$$

Donde:

- **AFP** es la medición inicial de puntos función de la Aplicación.
- **ADD** es la medición de puntos función no ajustados de aquellas funciones que fueron implantadas por el proyecto de desarrollo.
- **VAF** es el valor del factor de ajuste de la aplicación.

FÓRMULA PARA PROYECTOS DE MEJORA

La funcionalidad de la aplicación puede ser cambiada de varias formas:

- La **funcionalidad añadida** (nueva) aumenta el tamaño de la

aplicación.

- La **funcionalidad modificada** aumenta, disminuye o no cambia el tamaño de la aplicación.
- La **funcionalidad eliminada** disminuye el tamaño de la aplicación.
- Los cambios en el **valor del factor de ajuste** incrementan o disminuyen el tamaño de la aplicación.

Cualquier funcionalidad de conversión asociada a un proyecto de mejora se omite totalmente del cálculo de puntos función de la aplicación. Esto es debido a que el dato no modifica la funcionalidad de la aplicación sólo contabiliza los puntos función arrojados por el proyecto.

$$AFP = [(UFPB + ADD + CHGA) - (CHGB + DEL)] \cdot VAFA$$

Donde:

- **AFP** es la medición de puntos función ajustados de la Aplicación.
- **UFPB** es la medición de puntos función no ajustados de la aplicación antes de que el proyecto de mejora comience.
- **ADD** es la medición de puntos función no ajustados de aquellas funciones que fueron añadidas por el proyecto de mejora.
- **CHGA** es la medición de puntos función no ajustados de aquellas funciones que fueron modificadas por el proyecto de mejora. Este número refleja el tamaño de las funciones después de las modificaciones.
- **CHGB** es la medición de puntos función no ajustados de aquellas funciones que fueron modificadas por el proyecto de mejora. Este número refleja el tamaño de las funciones antes de hacer los cambios.
- **DEL** es la medición de puntos función no ajustados de aquellas funciones que fueron eliminadas por el proyecto de mejora.
- **VAFA** es el valor del factor de ajuste de la aplicación después de que el proyecto de mejora finalice.

CARACTERISTICAS GENERALES DEL SISTEMA

A continuación se presentan las 14 características generales del sistema que permiten calcular el Valor del Factor de Ajuste:

1. Comunicaciones de Datos

Las comunicaciones de datos se entienden como las que se establecen entre la aplicación y el procesador.

Los datos y la información de control se envían o reciben a través de utilidades de comunicación siguiendo protocolos determinados.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** La aplicación es un proceso batch puro o es una aplicación independiente.
- **1.** La aplicación es batch pero con entrada de datos o impresión remotas.
- **2.** La aplicación es batch pero tiene entrada de datos e impresión remotas.
- **3.** La aplicación tiene toma de datos online (interactiva) o un front-end de Teleproceso (TP) para un proceso batch o para una consulta.
- **4.** La aplicación es más que un front-end pero soporta únicamente un solo tipo de protocolo de comunicaciones de TP.
- **5.** La aplicación es más que un front-end y soporta más de un tipo de protocolo de comunicaciones de TP.

2. Procesamiento de Datos Distribuido

Esta característica intenta describir el grado de comunicación entre los distintos componentes físicos de la aplicación

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Los datos ni se transfieren ni se procesan en ningún otro componente del sistema.
- **1.** La aplicación prepara los datos para que se transfieran a otro componente del sistema y éste los procese, para tratamiento del usuario.
- **2.** La aplicación prepara los datos para que se transfieran a otro componente del sistema y éste los procese pero no es para tratamiento del usuario.
- **3.** El procesamiento y la transferencia de los datos son online (interactivos) y en una sola dirección.
- **4.** El procesamiento y la transferencia de los datos son online (interactivos) en ambas direcciones.

- **5.** El procesamiento y la transferencia de los datos son online (interactivos) y se realizan en el componente del sistema más adecuado para ello.

En este caso se entiende que la aplicación no reside únicamente en un componente hardware del sistema sino que sus datos se encuentran repartidos en distintos componentes hardware y que dichos componentes son reconocidos por el usuario.

3. Rendimiento

En esta característica se quiere describir la influencia que existe del tiempo de respuesta y la tasa de transferencia en las tareas de diseño de la aplicación.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** El usuario no especificó ningún requerimiento especial de rendimiento.
- **1.** Se establecieron y revisaron requerimientos de diseño y de rendimiento pero no se necesitaron acciones especiales.
- **2.** El tiempo de respuesta o la tasa de transferencia son críticos en las horas punta. No se necesitó ningún diseño especial para la utilización del procesador. La fecha límite del proceso es el siguiente ciclo de negocio.
- **3.** El tiempo de respuesta o la tasa de transferencia son críticos a lo largo de todo el tiempo de operación. No se necesitó ningún diseño especial para la utilización de la CPU. Los requerimientos de fecha límite son restrictivos con los procesos de interfaces.
- **4.** Además del anterior, los requerimientos de rendimiento del usuario debido a su rigurosidad requieren tareas de análisis de rendimiento en la fase de diseño.
- **5.** Además del anterior, en la fase de diseño, desarrollo o implantación se utilizaron herramientas de análisis de rendimiento para conseguir los requerimientos de rendimiento indicados por el usuario.

Casos frecuentes:

- De forma general las aplicaciones batch se puntúan de 0 a 4.
- Las aplicaciones on-line (interactivas) se puntúan de 0 a 4.
- Las aplicaciones basadas en Web se puntúan entre 4 ó 5.

4. Configuración Altamente Utilizada

Describe de qué forma han influenciado el desarrollo de la aplicación las restricciones de recursos.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No se incluyen restricciones de funcionamiento. Ni explicitas ni implícitas.
- **1.** Existen restricciones de funcionamiento pero no se necesita ningún esfuerzo especial para cumplirlas. Son menos restrictivas que para una aplicación típica.
- **2.** Existen restricciones de funcionamiento y se necesita un esfuerzo especial para cumplirlas mediante controladores o programas de control. Son las restricciones de una aplicación típica.
- **3.** Hay restricciones de funcionamiento que requieren limitaciones específicas sobre una parte de la aplicación en el procesador central o en un procesador dedicado.
- **4.** Las restricciones de funcionamiento establecidas requieren limitaciones especiales para toda la aplicación en el procesador central o en un procesador dedicado.
- **5.** Además, existen restricciones especiales para la aplicación en los componentes distribuidos del sistema.

5. Tasa de transacciones

Describe el grado en el cual ha influido, en el desarrollo de la aplicación, la tasa de transacciones.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No se prevé ningún periodo punta de transacciones.
- **1.** Tasa de transacciones baja que tiene un efecto mínimo sobre las fases de diseño, desarrollo e instalación.
- **2.** Tasa de transacciones media tiene algún efecto sobre las fases de diseño, desarrollo y/o instalación.
- **3.** Alta tasa de transacciones que afecta a las fases de diseño, desarrollo y/o instalación.
- **4.** Alta tasa de transacciones indicada por el usuario en los requisitos de la aplicación o en

los acuerdos a nivel de servicio tanto que requieren que se realicen tareas de análisis de rendimiento en la fase del diseño, desarrollo y/o instalación.

- **5.** Además del anterior, se requiere en las fases de diseño, desarrollo y/o implantación del uso de herramientas de análisis del rendimiento.

Casos frecuentes:

- Generalmente las aplicaciones Batch se puntúan de 0 a 3.
- Las aplicaciones online (interactivas) de 0 a 4.
- Los sistemas de tiempo real se puntúan de 0 a 5.

6. Entrada de Datos Online (interactiva)

Describe el grado en el que los datos son introducidos o recuperados a través de transacciones interactivas.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Todas las transacciones se procesarán de forma batch.
- **1.** Del 1% al 7% de las transacciones son interactivas.
- **2.** Del 8% al 15% de las transacciones son interactivas.
- **3.** Del 16% al 23% de las transacciones son interactivas.
- **4.** Del 24% al 30% de las transacciones son interactivas.
- **5.** Más del 30% de las transacciones son interactivas.

Casos frecuentes:

- Generalmente, las aplicaciones Batch puntúan entre 0 y 1.
- Los sistemas en tiempo real, telecomunicaciones y control de procesos se puntúan con un 5.
- La mayoría de las aplicaciones online (interactivas) se puntúan con un 5.

7. Eficiencia del Usuario Final

Describe el grado de consideración de los factores humanos y de la facilidad de uso para el usuario de la aplicación medida.

Se intenta evaluar un diseño orientado a la eficiencia del usuario final. El diseño incluye:

- Ayudas a la navegación: teclas de función, saltos, menús generados dinámicamente, hiperenlaces,…
- Menús.
- Ayuda y documentación online.
- Desplazamiento automático del cursor.
- Desplazamiento vertical u horizontal de líneas.
- Impresión remota (vía transacciones online).
- Teclas de función pre asignadas.
- Trabajos batch enviados desde transacciones online.
- Listas desplegables.
- Utilización intensa de video inverso, resaltado, colores, subrayados y otros indicadores.
- Impresión hard copy de documentación de transacciones online.
- Interfaz para ratón.
- Ventanas emergentes.
- Plantillas y/o valores por defecto.
- Soporte bilingüe (soporte de dos idiomas, se cuenta como cuatro ítems).
- Soporte multilingüe (soporte de más de dos idiomas, se cuenta como seis ítems).

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Ninguno de los ítems anteriores.
- **1.** De uno a tres de los ítems anteriores.
- **2.** De cuatro a cinco de los ítems anteriores.
- **3.** Seis o más de los ítems anteriores pero no existen requerimientos específicos del usuario en cuanto a eficiencia.
- **4.** Seis o más de los ítems anteriores y los requisitos establecidos para la eficiencia del usuario final son lo bastante fuertes como para requerir tareas de diseño para tener en cuenta

los factores humanos.

- **5.** Seis o más de los ítems anteriores y los requisitos establecidos para la eficiencia del usuario final son lo bastante fuertes como para requerir el uso de herramientas y procesos especiales para demostrar que se han conseguidos los objetivos.

Casos Frecuentes:

- Siempre que se despliegue en un entorno GUI se deberá asignar 4 ó 5.
- Normalmente sólo se asigna 5 a aplicaciones para el mercado general o para usuarios no técnicos y solamente si se tienen especialistas ergonómicos y/o estudios de usabilidad como parte de sus procesos.
- Las aplicaciones batch puras se puntúan con un 0.
- Las aplicaciones Web se puntúan con un 5 (requieren herramientas especiales para demostrar que se han alcanzado los objetivos).

8. Actualización online

Describe el grado de actualización online de los ficheros lógicos internos.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Ninguna.
- **1.** Se incluye la actualización online de uno a tres ficheros de control. El volumen de actualización es bajo y la recuperación fácil.
- **2.** Se incluye la actualización online de cuatro o más ficheros de control. El volumen de actualización es bajo y la recuperación fácil.
- **3.** Se incluye la actualización online de los principales ficheros lógicos internos.
- **4.** Además, la protección contra la pérdida de datos es esencial y ha sido especialmente diseñada y programada en el sistema.
- **5.** Además, los altos volúmenes conllevan consideraciones de coste en el proceso de recuperación. Se incluyen procedimientos de recuperación altamente automatizados con intervención mínima del operador.

Casos frecuentes:

- Las aplicaciones batch se puntúan con 0.
- La mayoría de las aplicaciones GUI se puntúan con 3 ó más.

9. Complejidad de Procesamiento

Describe el grado en el cual la lógica de proceso ha influido en el diseño de la aplicación. Se debe contabilizar si están presentes los siguientes componentes:

- Control sensible y/o proceso de seguridad específico de la aplicación.
- Lógica de proceso compleja.
- Proceso matemático complejo.
- Procesos con muchas excepciones, de manera que las transacciones incompletas deben ser procesadas de nuevo.
- Tratamiento complejo para manejar múltiples posibilidades de entrada/salida.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Ninguno de los componentes anteriores.
- **1.** Uno de los componentes anteriores.
- **2.** Dos de los componentes anteriores.
- **3.** Tres de los componentes anteriores.
- **4.** Cuatro de los componentes anteriores.
- **5.** Los cinco componentes anteriores.

10. Reusabilidad

Describe el grado en que la aplicación y el código de la misma han sido específicamente diseñados, desarrollados y mantenidos para ser utilizables en otras aplicaciones.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Código no reutilizable.
- **1.** Se emplea código reutilizable dentro de la aplicación.
- **2.** Menos del 10% del código de la aplicación pretende ser usado en más de una aplicación.
- **3.** El 10% o más del código de la aplicación pretende ser usado en más de una aplicación.
- **4.** La aplicación fue específicamente empaquetada y/o documentada para facilitar su reutilización y la aplicación está personalizada a nivel de código fuente.
- **5.** La aplicación fue específicamente empaquetada y/o documentada para facilitar su reutilización y la aplicación está personalizada para su uso mediante el mantenimiento de parámetros de usuario.

11. Facilidad de instalación

Describe el grado en que la conversión desde entornos previos ha influido en el desarrollo de la aplicación.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** El usuario no estableció consideraciones especiales y no se requiere ninguna preparación especial para la instalación.
- **1.** El usuario no estableció consideraciones especiales pero se requiere una preparación especial para la instalación.
- **2.** El usuario determinó requisitos de conversión e instalación y se proporcionaron y probaron guías de conversión e instalación. El impacto de la conversión en el proyecto no se considera importante.
- **3.** El usuario determinó requisitos de conversión e instalación y se proporcionaron y probaron guías de conversión e instalación. El impacto de la conversión en el proyecto se considera importante.
- **4.** Además del punto 2 se proporcionaron y probaron herramientas de conversión e instalación automáticas.
- **5.** Además del punto 3 se proporcionaron y probaron herramientas de conversión e instalación automáticas.

12. Facilidad de Operación

Describe el grado de desarrollo de la aplicación en aspectos de operación como el arranque, copias de seguridad y procesos de recuperación.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** El usuario no estableció ningún procedimiento especial de operación aparte de los procedimientos normales de copias de seguridad.
- **1-4.** Cada uno de los ítems siguiente tiene valor de 1 punto (salvo que se indique lo contrario):
    - Se proporcionaron procedimientos de arranque, de copia de seguridad y de recuperación pero se requiere la intervención humana.
    - Se proporcionaron procedimientos de arranque, de copia de seguridad y de recuperación y no se requiere la intervención humana (se cuenta como dos ítems).
    - La aplicación minimiza la necesidad de montar cintas y/o acceso de datos remotos siendo necesaria la intervención humana.
    - La aplicación minimiza la necesidad de manejo de papel.
- **5.** La aplicación está diseñada para una operación desatendida. No se necesita la intervención humana salvo para el arranque y la parada. La recuperación automática de errores forma parte de la aplicación.

13. Múltiples localizaciones.

Describe el grado de desarrollo de la aplicación para poder adaptarse a distintos entornos de hardware y software.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** En el diseño se consideran las necesidades de instalación de una sola localización.
- **1.** Se consideraron en el diseño de la aplicación las necesidades de más de una localización. La aplicación está diseñada para operar sólo bajo entornos de hardware y software idénticos.
- **2.** Se consideraron en el diseño de la aplicación las necesidades de más de una localización. La aplicación está diseñada para operar sólo bajo entornos de hardware y software similares.

- **3.** Se consideraron en el diseño de la aplicación las necesidades de más de una localización. La aplicación está diseñada para operar bajo entornos de hardware y software diferentes.
- **4.** Se proporcionan y prueban la documentación y el plan de mantenimiento para mantener la aplicación en localizaciones múltiples y la aplicación es como se describe en el punto 2.
- **5.** Se proporcionan y prueban la documentación y el plan de mantenimiento para mantener la aplicación en localizaciones múltiples y la aplicación es como se describe en el punto 3.

El concepto múltiples localizaciones no implica que tengan que ser hardware. Puede haber múltiples localizaciones en un mismo entorno físico determinándolo el que existan o no la necesidad de varias instalaciones.

14. Facilidad de Cambio

Describe el grado en que la aplicación ha sido desarrollada teniendo en cuenta facilitar el cambio de la lógica de negocio o de la estructura de datos.

Las siguientes características pueden ser aplicables al sistema:

A. Consulta flexible.

1. Se proporcionan consultas flexibles y facilidades de informes para realizar consultas simples (se cuenta como 1 ítem).
2. Se proporcionan consultas flexibles y facilidades de informes para realizar consultas de complejidad media (se cuenta como 2 ítems).
3. Se proporcionan consultas flexibles y facilidades de informes para realizar consultas complejas (se cuenta como 3 ítems).

B. Datos de Control del Negocio

1. Los datos de control del negocio se

guardan en tablas que mantiene el usuario con procesos interactivos online pero los cambios se hacen efectivos sólo en el siguiente ciclo de negocio (contar como 1 ítem).

2. Los datos de control del negocio se guardan en tablas que mantiene el usuario con procesos interactivos online y los cambios se hacen efectivos inmediatamente.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Ninguno de los ítems anteriores.
- **1.** Total de 1 ítem de los anteriores.
- **2.** Total de 2 ítems de los anteriores.
- **3.** Total de 3 ítems de los anteriores.
- **4.** Total de 4 ítems de los anteriores.
- **5.** Total de 5 ítems de los anteriores.

NESMA

NESMA es la *Netherlands Software Metrics Association*, en español Asociación Holandesa de Métricas del Software, así se conoce a la segunda asociación de medición funcional del software más grande del mundo.

Su sitio web es el siguiente: http://www.nesma.nl

ORIGEN

Esta asociación se fundó a finales de **1989** bajo el nombre de *NEFPUG* (Netherlands Function Point Users Group ó Grupo de usuarios holandeses de Puntos Función).

En **1991** *NEFPUG* forma un grupo de trabajo denominado "*FPA for Enhancement and Maintenance*" (Análisis en Puntos Función para Mejora y Mantenimiento) y en él desarrolló y publicó una serie de guías para la aplicación del FPA en las labores de Mejora y Mantenimiento del Software. La primera publicación se realizó en **1993**.

Originalmente, las guías estaban escritas en holandés y visto el interés internacional sobre el tema en cuestión, y la falta de documentación sobre el mismo, *NEFPUG* realizó la traducción de las guías al inglés.

En **1995**, *NEFPUG* cambió su nombre por el nombre actual: *NESMA*.

En **2008** se actualizó la documentación debido a la migración que había sufrido el manual de conteo de *NESMA* para poder adaptarlo a la certificación ISO del método (ISO 24570). Es el Definitions and counting guidelines for the application of function point analysis. NESMA Functional Size Measurement method compliant to ISO/IEC 24570 actualmente en la version 2.1 y también edita por separado su famoso *FPA for Software Enhancement* actualmente en la versión 2.2.1 (2009) donde expone sus recomendaciones para una medición de proyectos de mantenimiento o mejora.

INTRODUCCIÓN

Dentro de *NESMA* se percataron que la gran mayoría de usuarios del método de *IFPUG* lo utilizaban para contabilizar esfuerzos en los desarrollos.

Según el método de *IFPUG* el conteo en puntos función para una función creada es el mismo que para una función modificada, por eso en *NESMA* intentaron adaptar el método de *IFPUG* para que se ajustara mejor a la medición de proyectos donde se modificaba funcionalidad, es decir se adaptó a proyectos de mejora (mantenimiento) teniendo en cuenta únicamente los cambios realizados en las funciones transaccionales y en las funciones de datos.

Por ello, el método de *NESMA* contabiliza las funciones transaccionales o las funciones de datos de la siguiente forma:

- **Funciones nuevas.** Las funciones nuevas se contabilizan de la misma forma que en el método de *IFPUG*.
- **Funciones modificadas.** Las funciones existentes en el sistema pero que se ven alteradas por el proyecto que estamos llevando a cabo. Son las que se cuentan de una forma diferente aplicando el factor de impacto que veremos a continuación.
- **Funciones eliminadas.** Se les asigna un valor fijo de factor de impacto.

Cuando una función transaccional o de datos se modifica, el método de *NESMA* contabiliza el número de atributos y grupos lógicos cambiados en una y el número de atributos y subgrupos de atributos en la otra. A continuación y en función de ellos con unas tablas obtiene el % de impacto en la función.

El **factor de impacto** nos da una idea de cuál es el porcentaje de la función que estamos contabilizado que se ha visto afectado por la modificación. Por tanto, una función nueva tendrá un factor de impacto del 100% y una función modificada tendrá un valor menor que ese.

Como este porcentaje nos informa de que parte proporcional de la función transaccional o de datos está afectada por los cambios realizados, utilizándolo podremos calcular cuántos puntos función de la función completa son los que están afectados por la modificación.

CONCEPTOS

Mantenimiento, dentro de *NESMA* es entendido como todas aquellas actividades necesarias para sustentar la operación del sistema sin modificar su funcionalidad.

Mejora se entiende como los cambios en la funcionalidad de un sistema de información, es lo que comúnmente se denomina **mantenimiento adaptativo**. Otros tipos de mantenimiento como lo son correctivo, preventivo o perfectivo no se incluyen en el método porque no implican cambios en la funcionalidad del sistema.

Cuando una mejora resulta únicamente en la adición de funciones nuevas al sistema y no incluye cambios en funciones existentes, deben ser tratadas como nuevos desarrollos y ser medidas como el método *FPA* de *IFPUG* indica. El método de *NESMA* se aplica sólo a funciones existentes cuya funcionalidad se modifica.

El **Factor de Impacto (I)** es una medida del grado de cambio que sufre una función de datos o transaccional.

El tamaño funcional obtenido con el método de *NESMA* se mide en **EFP o Enhancement Function Points (Puntos Función de Mejora)** distintos de los *FP* del método *FPA* de *IFPUG*, son una unidad de medida diferente.

MÉTODO

El método tiene dos partes diferenciadas. La primera de ellas se encarga de la medición de los **EFP** debido a modificaciones en la funcionalidad del sistema y la segunda parte se encarga de medir los **TFP** (*Test Function Points* ó Puntos Función de Pruebas) que miden la funcionalidad que no ha sufrido ningún cambio debido a los requerimientos de la mejora/modificación del sistema pero que hay que probar para comprobar que siguen funcionando de la forma esperada.

MIDIENDO LOS EFP

La primera parte del método, encargada de realizar la medición funcional de las mejoras introducidas en el sistema, consta de los siguientes pasos:

Paso 1. Identificación de las funciones a medir

Utilizando el método *FPA* de *IFPUG* se determinará cuales con las funciones de datos y transaccionales que se encuentran dentro del ámbito de la mejora del proyecto, es decir se encuentran impactadas por él. Es deseable que exista un conteo en *PF* según *IFPUG* del sistema previo al conteo de la mejora. Si no existe como mínimo debe existir un conteo de las funciones impactadas.

Paso 2. Determinar las funciones añadidas

Se identificarán las funciones que se están añadiendo al sistema (antes no existían) con el proyecto de mejora. Dichas funciones se medirán siguiendo el método estándar de *FPA* de *IFPUG*. El factor de impacto para estas funciones será 1.00.

$$EFP_{ADDED} = FP_{ADDED}$$

Paso 3. Determinar las funciones que se eliminan

Se identifican las funciones que serán eliminadas por el proyecto de mejora y se calcula su tamaño según el método *FPA* de *IFPUG*.

A estas funciones se les aplica un factor de impacto fijo de 0.40.

$$EFP_{DELETED} = FP_{DELETED} \cdot 0.40$$

Paso 4. Determinar las funciones de datos modificadas y calcular el factor de impacto de las mismas

Para cada función de datos habrá que determinar si ha sufrido cambios en los siguientes aspectos:

- La función de datos sufre un cambio interno. Se han añadido, eliminado o modificados DET.

El porcentaje de cambio en este caso se calcula de la siguiente forma:

$$\text{Porcentaje de cambio} = \left(\frac{\text{Nº de DETs añadidos} \diagup \text{modificados} \diagup \text{eliminados}}{\text{Nº de DETs de la función de datos original}} \right) \cdot 100$$

Una vez obtenido el porcentaje de cambio de la siguiente tabla podemos saber el factor de impacto (ver Tabla).

Porcentaje DETs	≤1/3 (x 100%)	≤2/3 (x 100%)	≤100%	> 100%
Factor de Impacto	0,25	0,50	0,75	1,00

Tabla - Cálculo del factor de Impacto en una función de Datos en el método NESMA

- La función de datos no sufre un cambio interno sino que cambia el tipo. Por ejemplo:

De EIF pasa a ILF o al revés.

En este caso se le asigna a la función un factor de impacto de 0.40.

Si una función de datos es a la vez modificada por un cambio interno y por un cambio no interno, se debe calcular el primero y nos quedaremos con aquel de los dos que sea más alto.

El número de puntos función de mejora de la función de datos es determinado de la siguiente forma:

$$\mathrm{EFP}_{\mathrm{CHANGED}} = \mathrm{FP}_{\mathrm{CHANGED}} \cdot I_{\mathrm{CHANGED}}$$

Si se diera el caso en el cual un EIF ó un ILF es dividido en dos o más funciones, se deberán contar como una función eliminada y dos o más creadas.

Si se diera el caso en el cual un EIF y un ILF son combinados, se deberán contar dos eliminados y uno nuevo añadido.

Paso 5. Determinar las funciones transaccionales modificadas y calcular el factor de impacto de las mismas

Una vez identificadas las funciones incluidas dentro del ámbito del proyecto de mejora, se debe calcular el tamaño funcional de cada una de ellas después de la modificación.

Una función transaccional se considera que ha sido cambiada si se ha

alterado de cualquier forma pero sigue manteniendo el mismo nombre y propósito después de la mejora.

Una función puede estar afectada por un cambio directo recogido en los requerimientos de usuario o por un cambio indirecto de una funcionalidad que afecta internamente a la misma.

Una función será modificada si sucede al menos una de las siguientes situaciones:

- Se han añadido, modificado o eliminado DET.
- Se han añadido, modificado o eliminado ficheros lógicos de datos (ILF o EIF)
- El interfaz de usuario ha sido cambiado funcionalmente.
- La lógica de negocio que se incluye en una función transaccional ha sido cambiada.
- Se ha realizado un cambio cosmético visible por el usuario.

Los cambios en los nombres de los DET no se contabilizan como cambios en la función transaccional, para tenerlos en cuenta como cambio debe modificarse la naturaleza del DET.

Para calcular los EFP de una función transaccional procederemos con los siguientes 4 pasos:

a) Identificamos los DET y FTR usados por la función transaccional.
En este caso, si la función transaccional únicamente sufre un cambio cosmético los DET y FTR afectados serán 0, por ello se toma como impacto el mínimo: 0.25.

b) Cálculo del porcentaje de DET y FTR cambiados durante la mejora.
El factor de impacto se calcula a través del porcentaje de cambio sufrido

en el número de DET y FTR con respecto a la función original.

$$\text{Porcentaje DETs} = \left(\frac{\text{Nº de DETs añadidos} / \text{modificados} / \text{eliminados}}{\text{Nº de DETs de la función transaccional original}} \right) \cdot 100$$

$$\text{Porcentaje FTRs} = \left(\frac{\text{Nº de FTRs añadidos} / \text{modificados} / \text{eliminados}}{\text{Nº de FTRs de la función transaccional original}} \right) \cdot 100$$

c) Cálculo del factor de impacto de la función transaccional

Se calcula comparando los valores obtenidos en el paso anterior en la siguiente tabla (ver Tabla):

Factor de Impacto de Función Transaccional		Porcentaje DETs		
		≤⅔(x 100%)	≤100%	> 100%
Porcentaje FTRs	≤⅓(x 100%)	0,25	0,50	0,75
	≤⅔(x 100%)	0,50	0,75	1,00
	≤100%	0,75	1,00	1,25
	> 100%	1,00	1,25	1,50

Tabla - Cálculo del factor de impacto de una función transaccional en el método NESMA

Cuando el factor de impacto excede el 1.00 se debe revisar si tiene sentido o no.

d) Calcular el tamaño funcional en EFP

Se calcula aplicando la siguiente fórmula:

$$EFP_{CHANGED} = FP_{CHANGED} \cdot I_{CHANGED}$$

Paso 6. Calcular el tamaño del proyecto de mejora

El tamaño funcional del proyecto de mejora se tendrá con la siguiente fórmula:

$$EFP_{TOTAL} = \sum EFP_{AÑADIDOS} + \sum EFP_{ELIMINADOS} + \sum EFP_{MODIFICADOS}$$

Paso 7. Calcular el tamaño del sistema después de la mejora

El tamaño del sistema después de la mejora se obtiene como sigue:

$$FP_{NUEVO} = FP_{BASE} + \sum FP_{AÑADIDOS} + (\sum FP_{DESPUES-CAMBIO} - \sum FP_{ANTES-CAMBIO}) - \sum FP_{ELIMINADOS}$$

En este caso, todos los Puntos Función calculados son sin tener en cuenta ningún factor de impacto siguiendo el método *FPA* de *IFPUG*.

MIDIENDO LOS TFP

La segunda parte se encarga de realizar la medición de los puntos función de Pruebas.

Generalmente, el rango de funciones de datos y transaccionales que hay que probar debido a un proyecto de mejora es mucho más amplio que el rango de

funciones a modificar.

El tamaño de las funciones a ser probadas se mide en ***TFP***, *Test Function Points* o *Puntos Función de Pruebas*. Cuando estamos realizando el cálculo de los *TFP* no se tiene en consideración el Factor de Impacto. Debemos tener en cuenta si la función es añadida o modificada ya que en este caso si una función ha sido eliminada no tiene sentido que se pruebe.

El tamaño se mide siguiendo el estándar *FPA* de *IFPUG* y el total será la suma de todos los *TFP* de las funciones detectadas.

Las funciones modificadas siempre se contarán después de la modificación.

$$TFP = \sum FP_{PRUEBAS}$$

APLICANDO EL MÉTODO

Cuando se estima utilizando esta técnica, se pueden presentar desviaciones apreciables en casos concretos pero si miramos la media, las estimaciones son correctas.

Los EFP y los TFP se pueden utilizar para generar métricas de productividad como horas por EFP u horas por TFP. Estos valores difieren de las horas por punto función calculados con *FPA* de *IFPUG*.

El esfuerzo total de mejora, incluyendo pruebas, puede ser expresado de la siguiente forma:

$$\text{Esfuerzo Total Mejora} = (\text{N° de EFPs} \cdot \text{horas por EFP}) + (\text{N° de TFPs} \cdot \text{horas por TFP})$$

MKII-FPA

UKSMA Son las siglas de la United Kingdom Software Metrics Association o Asociación de Métricas del Software del Reino Unido.

Fue fundada en **1998**, originalmente conocida como la asociación MkII FP User Group (o Grupo de Usuarios de Puntos Función MKII).

La dirección de su sitio Web es http://www.uksma.co.uk/

ORIGEN

MK-II o Mark II es un método que fue desarrollado por *KPMG* entre **1985** y **1986** como un método propietario.

La primera definición del mismo la realizó *Charles Symons* en **1988**, junto con una serie de comentarios sobre el método de *Albrecht*, en el documento "*Function Point Analysis: Difficulties And Improvements*", *IEEE TTransactions on Software Engineering, SE-14(1).*

Finalmente, definido por *Charles Symons* en **1991** con la publicación de "*Software Sizing and Estimation: Mk II FPA"* pasó a ser un método público.

La última versión del manual es la 1.3.1 realizada en **1998**.

Se aprobó como estándar ISO en **2002**: *ISO/IEC 20968:2002*.

Este método fue adoptado por la UKSMA como su principal método a la hora de obtener la medición funcional del software.

INTRODUCCIÓN

El método Mk-II FPA distingue dos tipos de entidades:

- **Entidades Primarias.** Estás entidades son aquellas por las cuales

hemos diseñado el sistema, para almacenarlas. Por ejemplo : *Empleado, Contrato, Factura,…*

- **Entidades No Principales.** El resto de entidades cuyo objetivo es servir de valores de referencia, validación, etc. y que generalmente suelen constar de un código, descripción y/o conjunto de valores válidos.

Una **Transacción Lógica**, es una combinación de Entrada, Proceso y Salida, correspondientes todos a un único proceso desde un punto de vista lógico y desencadenada por un evento de interés o por una necesidad de usuario.

En el método de MKII FPA se miden dos componentes del software:

- **Componente funcional.** Es la suma de los tamaños de todas las transacciones lógicas que se identifiquen dentro del alcance.
- **Componente técnico.** Que consiste en la evaluación de las características generales de la aplicación (de forma similar a como explicábamos para el método de IFPUG). Esta parte es opcional y generalmente no se utiliza en las mediciones.

El tamaño de una **Transacción Lógica** es igual a la suma de los elementos que la forman:

- **Elementos de Entrada.** Serían todos los atributos que recibe como entrada la transacción.

- **Elementos de Proceso.** Serían todas las entidades principales a las que se referencia: se leen o se actualizan.

Cada **Entidad Primaria** se contabiliza una única vez sin importar las veces que se referencie, salvo que se relacione consigo misma.

La referencia a las **Entidades No Principales**, no importa el número de ellas al que se acceda, se contabilizará como un acceso a una entidad genérica denominada *Entidad del Sistema*.

- **Elementos de Salida.** Serían todos los atributos que se muestran en la transacción.

A diferencia del método de IFPUG si un Elemento de Datos aparece en la Entrada y en la Salida, en el método Mk-II FPA debe contarse dos veces.

Para medir el componente técnico, como decíamos anteriormente, deberemos evaluar de 0 a 5 las características del sistema. Éstas son las siguientes:

1. **Comunicación de Datos.**
2. **Función Distribuida.**
3. **Rendimiento.**
4. **Configuración fuertemente utilizada.**
5. **Tasa de Transacciones.**
6. **Entradas de datos on-line (interactiva).**
7. **Diseño teniendo en cuenta la Usabilidad.**
8. **Actualización on-line (interactiva).**
9. **Complejidad de Proceso.**
10. **Reusable en otras aplicaciones.**
11. **Facilidad de Instalación.**
12. **Facilidad de Operación.**
13. **Múltiples localizaciones.**
14. **Facilidad de Cambio.**
15. **Requerimientos de otras aplicaciones.**
16. **Seguridad, Privacidad, Auditoria.**
17. **Necesidades de formación de los usuarios.**
18. **Utilización directa por otras empresas.**
19. **Documentación.**
20. **Características definidas por el usuario.**

La última de ella nos permite añadir a las 19 anteriores aquellas características que estimemos que son importantes para nuestros sistemas.

CONCEPTOS

Una **Entidad** se entiende como cualquier elemento del mundo real acerca del cual queremos conocer información. Las ocurrencias de una entidad se pueden identificar individualmente o colectivamente, por ejemplo un Empleado y todos los Empleados de la Empresa, pudiendo tener en cada caso atributos de interés diferentes.

Algunas entidades, por su naturaleza, sólo tendrán interés en una faceta específica del sistema (o en varias) debiéndose evaluar únicamente en aquella en la cual tenga relevancia y no en las otras, por ejemplo una entidad sólo tiene sentido en la Salida pero no en almacenar sus ocurrencias, sólo se tendrá en cuenta en el componente de Salida.

Las **Entidades Primarias** son aquellas para las cuales el sistema ha sido diseñado para almacenar y mantener sus ocurrencias, por ejemplo:

Empleado, Contrato, Factura, Cliente, Consumo,…

El resto de Entidades cuyo objetivo es servir de valores de referencia, validación, adaptación del sistema a distintas circunstancias, etc. se denominan **Entidades no Principales**. Dichas entidades, normalmente, suelen constar de Código y Descripción ó de conjunto de valores válidos.

Una **Transacción Lógica** es una combinación de uno o más tipos de Entrada, algún procesamiento y uno o más tipos de salida correspondientes todos a un proceso lógicamente único (un proceso único desde un punto de vista lógico).

Una **Transacción** está desencadenada por un evento de interés o por una consulta/extracción de información del usuario que no afecta a los datos almacenados.

Si un proceso interno aparece en más de un transacción deberá contarse en cada

una de las transacciones que aparezca. No debe contarse como una transacción diferente a menos que sea desencadenado por un evento de significación para el usuario.

MÉTODO

El proceso de cálculo del tamaño funcional del software es muy similar al proceso realizado en el método de IFPUG. Por una parte, se evalúan las transacciones lógicas del software (componente funcional) y, por otro lado, las características generales de la aplicación (componente técnico). En este caso, las características generales son 5 más de partida y se pueden extender, si así se estima oportuno, en el análisis del sistema.

Para la medición de la funcionalidad, la unidad es la transacción lógica entendida como una combinación de Entrada, Proceso y Salida, desencadenada por un evento de interés o por una necesidad del usuario.

El valor de la funcionalidad será el sumatorio ponderado de estos tres componentes medidos en el sistema:

- En el **componente de entrada** de una transacción, contabilizamos todos los tipos de Elementos de Dato que entran en la misma (cruzan la frontera).
- En el **componente de proceso** de una transacción, contabilizamos todas las entidades principales a las cuales se referencia.
- En el **componente de salida** de una transacción, se contabilizan todos los tipos de Elementos de Datos que se muestran (cruzan la frontera).

A la hora de medir las transacciones del sistema objeto de estudio, debemos contabilizar el número de referencias a *Entidades Principales* que existan. Si una

Entidad Principal se lee y se escribe en una misma transacción sólo se tendrá en cuenta una única vez.

En el caso de que una *Entidad Principal* se relacione consigo misma, deberá contabilizarse como dos ocurrencias diferentes.

Cuando en el sistema se hace referencia a una Entidad no Primaria, se entenderá que se está accediendo a una Entidad Genérica denominada *Entidad del Sistema*.

Una *Entidad Primaria* puede estar compuesta de subentidades que podrán ser contabilizadas como independientes entre sí para una transacción si el procesamiento que lleva aparejado cada una de ellas o los atributos que se referencian son diferentes.

A la hora de contabilizar los elementos de datos se deberá tener en cuenta que siempre se están contabilizando tipos de datos y no ocurrencias de los mismos.

Todos los mensajes de error que puedan ser lanzados por una aplicación se contabilizarán como ocurrencias del mismo *Elemento de Datos* "mensaje de error".

De igual forma, el resto de mensajes que no sean de error se contabilizarán como ocurrencias del *Elemento de Datos* "mensaje de operador".

Cuando se estén contabilizando tablas se tendrán en cuenta los tipos de datos tanto para filas como para columnas y no así las ocurrencias de los mismos. El total de la tabla será el total de Elementos de Datos de las columnas por el total de Elementos de Datos de las filas. Si filas distintas tienen distinto procesamiento, se contabilizaran como diferentes. Si en las columnas se muestran totalizadores para distintas agregaciones, se contabilizarán como diferentes.

Si un Elemento de Dato de entrada se vuelve a mostrar en la salida se debe contar dos veces. Esto es así salvo que se muestre en la salida debido a un error de validación que entonces se tomará que al mostrarlo en la salida es una ocurrencia del elemento de datos “mensaje de error” ya que la transacción todavía se encontraría en la fase de entrada.

Sumando, para todo el sistema, cada una de las componentes de Entrada, Proceso y Salida el valor global lo obtenemos de la siguiente forma:

$$\mathrm{UFP} = W_I \cdot N_I + W_E \cdot N_E + W_O \cdot N_O$$

Donde:

- $\mathbf{N_I}$ Valor del Componente de Entrada.
- $\mathbf{N_E}$ Valor del Componente de Entidad Referenciada (Proceso).
- $\mathbf{N_O}$ Valor del Componente de salida
- $\mathbf{W_I}$ Peso para los componentes de entrada
- $\mathbf{W_E}$ Peso para los componentes de entidad referenciada (Proceso)
- $\mathbf{W_O}$ Peso para los componentes de salida
- **UFP** Unajusted Function Points. Nos da el valor en Puntos Función MKII sin ajustar.

El valor de los pesos para los distintos componentes, puede ser calibrado por nosotros para nuestro entorno para un mejor ajuste o podemos utilizar los valores estándares del mercado que son:

- $W_I = 0.58$
- $W_E = 1.66$
- $W_O = 0.26$

Para poder calcular el valor de los puntos función ajustados primero debemos realizar el ajuste de la complejidad técnica determinando el valor de las

características del sistema.

CARACTERÍSTICAS DEL SISTEMA

Los valores de las características se puntúan de 0 (Sin influencia) a 5 (Fuerte influencia).

1. Comunicaciones de Datos

Uso de la infraestructura de comunicación para los Datos y para la información de control

Para los terminales conectados remota o localmente se puntuará de la siguiente forma:

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Procesamiento Batch puro o funcionamiento independiente de un PC.
- **1.** Batch con entrada de datos o impresión remota.
- **2.** Batch con entrada de datos e impresión remota.
- **3.** Colección de datos online (interactiva) front-end TP (Teleproceso).
- **4.** Más que front-end pero la aplicación soporta únicamente 1 protocolo de comunicaciones TP.
- **5.** Más que un front-end, la aplicación soporta más de 1 protocolo de comunicaciones TP .

2. Función Distribuida

La aplicación se extiende sobre 2 o más procesadores.

Datos distribuidos o procesamiento incorporado dentro de la aplicación, se puntuará de la siguiente forma:

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No hay facilidades para transferir datos o funciones de proceso entre componentes de la

aplicación.

- **1.** La aplicación prepara datos para el procesamiento del usuario final en otro componente de la aplicación.
- **2.** La aplicación prepara datos para procesamiento en otra máquina (no usuario final).
- **3.** Procesamiento distribuido, on-line(interactivo), las transferencias de datos se realizan en un sentido únicamente.
- **4.** Procesamiento distribuido, on-line (interactivo), transferencias de datos en ambos sentidos.
- **5.** Funciones de procesado son dinámicamente ejecutadas en el componente de la aplicación más apropiado para ello.

3. Rendimiento

Influencia en la aplicación de la respuesta de la misma y del rendimiento.

Puntuar como influyen los objetivos de rendimiento de la aplicación acordados con el usuario en el diseño, desarrollo, instalación y soporte de la misma:

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No se han definido requerimientos especiales.
- **1.** Requerimientos de rendimiento indicados y revisados pero no han sido necesarias acciones especiales.
- **2.** Respuesta on-line (interactiva) critica durante las horas pico. No hay un diseño especial del uso de la CPU.
- **3.** Respuesta on-line (interactiva) crítica durante el día laboral. No hay un diseño especial de uso de la CPU. El intervalo de procesamiento afectado por las aplicaciones de interfaces.
- **4.** Los requerimientos de rendimiento necesitan un análisis de rendimiento durante el diseño.
- **5.** Herramientas de análisis de rendimiento se utilizaron en el diseño, desarrollo y/o fase de instalación para cumplir los requerimientos de rendimiento indicados por el usuario.

4. Configuración Fuertemente Utilizada

La aplicación está fuertemente ligada a la configuración, al equipamiento.

La configuración destino es fuertemente utilizada y se puntúa la necesidad de tenerla en cuenta durante el desarrollo:

Guía de Aplicación (se puntúa de 0 a 5):

- **0-3.** Aplicación típica en una máquina de producción estándar. No se indican restricciones de operación.
- **4.** Las restricciones de operación indicadas requieren restricciones especiales en la aplicación en el único procesador central.
- **5.** Además, hay restricciones especiales en la aplicación en sus componentes distribuidos.

5. Tasa de Transacciones

Un pico de alta tasa de transacciones puede causar problemas diferentes a los causados por la característica 3.

El ratio de transacciones tiene influencia en el diseño, desarrollo, instalación y soporte de la aplicación. Puntuar de la siguiente manera:

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No hay previsto un periodo de transacciones máxima (pico).
- **1.** 10% de las transacciones están afectadas por el tráfico máximo.
- **2.** 50% de las transacciones están afectadas por el tráfico máximo.
- **3.** Todas las transacciones están afectadas por el tráfico máximo.
- **4.** Análisis de rendimiento en la fase de diseño.
- **5.** Herramientas de análisis de rendimiento utilizadas en el diseño, desarrollo e instalación.

6. Entrada de datos interactiva (on-line)

El terminal es utilizado como entrada.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Todas las transacciones son batch.
- **1.** Del 1 al 7% de las transacciones son interactivas.
- **2.** Del 8 al 15% de las transacciones son interactivas.
- **3.** Del 16 al 23% de las transacciones son interactivas.
- **4.** Del 24 al 30 % de las transacciones son interactivas.
- **5.** Más del 30% de las transacciones son interactivas.

7. Diseño teniendo en cuenta la Usabilidad

Los factores humanos son tenidos en cuenta en el diseño. El usuario requiere funciones on-line (interactivas) para ayudar en el uso de la aplicación:

- Ayudas a la navegación
- Menús
- Ayuda interactiva
- Movimiento automático del cursor
- Scroll
- Impresión remota
- Teclas de función pre asignadas (atajos de teclado)
- Trabajos batch desde transacciones on-line (interactivas)
- Selección con el cursor de datos en pantalla
- Alta utilización del equipamiento del monitor (color, intensidad, etc.)
- Hard copy de transacciones interactivas
- Ratón
- Ventanas
- Pantallas mínimas
- Bilingüe
- Multidioma.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Ninguna de las opciones de arriba.
- **1.** 1-3 de las opciones de arriba.
- **2.** 4-5 de las opciones de arriba.
- **3.** más de 6 de las opciones de arriba.

- **4.** Además de la anterior existen tareas de diseño relativas a factores humanos.
- **5.** Además de la anterior, se utilizan herramientas especiales y prototipado para demostrar que los objetivos de usabilidad se han alcanzado.

8. Actualización interactiva (on-line)

Actualización de datos en tiempo real.

Los datos son actualizados en tiempo real:

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Nada.
- **1-2.** Actualización on-line (interactiva) de ficheros de control. El volumen de la actualización es bajo y la recuperación es fácil.
- **3.** Actualización on-line (interactiva) de la mayor parte de los ficheros lógicos internos.
- **4.** Además, la protección contra la pérdida de datos es esencial.
- **5.** A demás, el alto volumen aporta consideraciones de coste en el proceso de recuperación.

9. Complejidad del Proceso

Complejidad interna más allá de lo tratado por las convenciones de conteo de entidades de MKII.

Cuales aplican de las siguientes casuísticas:

- Control de sensibilidad (por ejemplo proceso especial de auditoría) y/o procesamiento de seguridad especifico para la aplicación
- Procesamiento lógico extensivo
- Procesamiento matemático extensivo.
- Mucho procesamiento de excepciones, muchas transacciones incompletas y mucho reproceso de transacciones.
- Procesamiento complejo para manejar múltiples posibilidades de Entrada/Salida (por ejemplo, multimedia, independencia de dispositivos)

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No aplica ninguna.
- **1.** Aplica una de las de arriba.
- **2.** Aplican dos de las de arriba.
- **3.** Aplican tres de las de arriba.
- **4.** Aplican cuatro de las de arriba.
- **5.** Aplican todas las de arriba.

10. Reusable en otras Aplicaciones

El código ha sido diseñado para ser compartido o utilizado por otra aplicación. No confundir con el factor 13.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No hay código reusable.
- **1.** El código reusable se utiliza dentro de la propia aplicación.
- **2.** Menos del 10% de la aplicación se considera reusable.
- **3.** Más del 10% de la aplicación se considera reusable.
- **4.** La aplicación fue específicamente empaquetada / documentada para ayudar a su reutilización y fue prepara hasta nivel del código fuente.
- **5.** La aplicación fue específicamente empaquetada / documentada para ayudar a su reutilización y fue prepara a través de parámetros.

11. Facilidad de Instalación

Si la conversión de los datos o la facilidad de instalación se tuvieron que tener en cuenta en la fase de diseño.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** El usuario no indicó consideraciones especiales para la conversión o para la instalación.
- **1.** El usuario no indicó consideraciones especiales para la conversión o para la instalación pero si se requiere una configuración especial.

- **2.** El usuario indica requerimientos de conversión e instalación y se han proveído y probado guías de instalación.
- **3.** Además de lo indicado en el punto anterior, el impacto de la configuración es considerado importante.
- **4.** Además de lo indicado en el punto 2, se han proveído y probado herramientas de conversión e instalación.
- **5.** Además de lo indicado en el punto 3, se han proveído y probado herramientas de conversión e instalación.

12. Facilidad de Operación

Facilidades de operación consideradas en el diseño de la aplicación.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** El usuario no ha indicado consideraciones operacionales especiales.
- **1-4.** Seleccionar de los siguientes, cada uno tiene 1 punto de valor salvo que se indique lo contrario:
    - Se requieren procedimientos de inicio específico de la aplicación, de backup y de recuperación. Todos ellos se proveen y se han probado pero es necesaria la intervención de un operario.
    - Igual al de arriba pero no requiere la intervención del operario (2 puntos).
    - La aplicación minimiza la necesidad de montar cinta.
    - La aplicación minimiza la necesidad manipulación de papel.
- **5.** La aplicación ha sido diseñada para la operación desatendida, entendiendo por ello que no es necesaria la intervención del operador salvo para el inicio de la aplicación o del apagado de la misma. La recuperación de los errores es automática.

13. Múltiples Localizaciones

La aplicación está diseñada para ser utilizada en múltiples localizaciones y/o múltiples organizaciones.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No hay requerimientos del usuario para considerar que necesita más de una localización.
- **1.** En el diseño se ha tenido en cuenta múltiples localizaciones pero idénticas en software y hardware.
- **2.** En el diseño se ha tenido en cuenta múltiples localizaciones pero similares en software y hardware.
- **3.** En el diseño se ha tenido en cuenta múltiples localizaciones, el software y el hardware pueden ser diferentes.
- **4-5.** Añadir 1 punto por cada uno de los siguientes:
  - Se proveen y se prueban la documentación y los planes de soporte para mantener la aplicación en múltiples localizaciones.
  - Las localizaciones están en diferentes países.

14. Facilidad de Cambio

Consideración en el diseño de futuros cambios en la aplicación. La aplicación ha sido específicamente diseñada y desarrollada para facilitar los cambios futuros en los requerimientos de usuario.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No hay requerimientos especiales de diseño de la aplicación para minimizar o facilitar el cambio.
- **1.** Se provee la capacidad de poder realizar queries de forma flexible para poder satisfacer solicitudes lógicas simples.
- **2.** Se provee la capacidad de poder realizar queries de forma flexible para poder satisfacer solicitudes lógicas de tipo medio.
- **3.** Se provee la capacidad de poder realizar queries de forma flexible para poder satisfacer solicitudes lógicas complejas.
- **4-5.** Sumar puntos de la siguiente manera:
  - Añadir 1 punto si datos de control significativos son guardados en tablas mantenidas a través de procesos online (interactivos) con realización de cambios en un momento posterior.
  - Añadir 2 puntos si datos de control significativos son guardados en tablas mantenidas con actualización interactiva on-line.

15. Requerimientos de Otras Aplicaciones

Interfaces.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** La aplicación es totalmente independiente.
- **1-5.** Los requerimientos de la aplicación para interfaces o intercambio de datos deben estar sincronizados con otras aplicaciones. Se debe contar 1 por cada aplicación hasta 5 como máximo

16. Seguridad, Privacidad, Auditoria

Características especiales de confidencialidad/seguridad. Se deben añadir a la puntuación los siguientes valores que sean relevantes.

Guía de Aplicación (se puntúa de 0 a 5):

- **0-5.** Sumar puntos de la siguiente manera:
  - 1 punto si la aplicación debe cumplir requerimientos de privacidad relativos a personas, posiblemente legales.
  - 1 punto si la aplicación debe cumplir requerimientos especiales de auditoría.
  - 2 puntos si la aplicación debe cumplir requerimientos excepcionales de seguridad.
  - 1 punto si requiere la encriptación de las comunicaciones de datos.

17. Necesidades de Formación de los Usuarios

Requerimientos específicos.

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Si no se han desarrollado materiales o cursos especiales de formación.
- **1.** Se provee un tutorial de ayuda estándar.
- **2.** Se provee un tutorial de ayuda de tipo Hipertexto.
- **3.** Se provee material para el curso de formación.
- **4.** Se provee material para curso de formación on-line .

- **5.** Se tienen requerimientos para una aplicación completa por separado o un simulador con propósito de ayudar en las tareas de formación.

18. Utilización Directa por Otras Empresas

Grado de uso/conexión de la aplicación

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** No hay conexión de otras empresas a la aplicación.
- **1.** Se envían o reciben datos a/de otras empresas que no se conocen.
- **2.** Otras empresas que son conocidas se conectan directamente a la aplicación en modo de sólo lectura.
- **3.** Otras empresas que son conocidas se conectan directamente a la aplicación con capacidad de modificación interactiva (on-line).
- **4.** Otras empresas que son conocidas se conectan directamente a la aplicación con capacidad interactiva (on-line) para crear, actualizar y eliminar.
- **5.** Otras empresas que no se conocen (público en general o tan amplio que no puede ser formado específicamente) pueden acceder a la aplicación .

19. Documentación

Contar uno por cada tipo de documento listado abajo que es distribuido y actualizado al final del proyecto:

- Especificación Funcional (proceso y datos)
- Especificación técnica.
- Documentación de programa (al menos diagramas de flujo)
- Biblioteca de Elementos de Datos
- Referencias Cruzadas de Elementos de datos, Registro y Programa.
- Manual de Usuario
- Manual de operaciones
- Descripción general de la aplicación
- Biblioteca de datos de prueba

- Materiales de Curso de Formación de Usuarios
- Documento de seguimiento del Coste/Beneficio de la aplicación
- Log del informe de Errores/Solicitud de Cambio

Guía de Aplicación (se puntúa de 0 a 5):

- **0.** Si tiene entre 0 y 2 tipos de documentos.
- **1.** Si tiene entre 3 y 4 tipos de documentos
- **2.** Si tiene entre 5 y 6 tipos de documentos
- **3.** Si tiene entre 7 y 8 tipos de documentos
- **4.** Si tiene entre 9 y 10 tipos de documentos
- **5.** Si tiene entre 11 y 12 tipos de documentos

XX. Características Definidas por el Usuario

Se pueden añadir a las 19 características anteriores aquellas que por los requerimientos de usuario se estimen oportunas.

CÁLCULO AJUSTADO

Una vez obtenidos los valores de las características, el cálculo del **TCA** (Technical Complexity Adjustment) se realiza con la siguiente fórmula:

$$\mathrm{TCA} = 0.65 + W \cdot \sum_{i=1}^{20} C_i$$

Donde:

- **W** es un valor estándar de la industria valorado en 0.005
- $\mathbf{C_i}$ es el valor de cada una de las características incluidas las definidas por el usuario.

Finalmente, el valor de los puntos de función ajustados del sistema los obtendremos multiplicando los puntos de función no ajustados por el TCA.

COSMIC FFP

Desarrollado por el *Common Software Metrics Internacional Consortium*, el denominado método *Full Function Points* o en general COSMIC-FFP o COSMIC, es considerado como el primer método de medición funcional de segunda generación.

Su sitio web es: http://www.cosmicon.com

ORIGEN

Inicialmente, el método fue desarrollado en **1997** por un equipo de la Universidad de Québec encabezado por el profesor *Denis St-Pierre,* con el fin de ampliar el dominio funcional del método de puntos función hacia entornos en tiempo real.

En **1999**, el *Common Software Metrics Internacional Consortium (Consorcio Internacional de Métricas del Software Comunes)* encabezado por *Alain Abran* y *Charles Symons* publican la primera versión del manual de conteo del COSMIC-FFP (la numeración de la versión es la 2.0).

En **2001** se publica la versión 2.1, con una serie de aclaraciones sobre conceptos de la primera versión.

En **2003** se publica la versión 2.2, reconocida como estándar con la norma ISO/IEC 19761.

En **2009** se publica la penúltima versión, la versión 3.0.1, que mantiene las mismas reglas de medición pero clarifica algunos conceptos.

En **2014** se publica la última versión hasta la fecha, la versión 4.0. Supone un gran cambio con respecto a la versión anterior pero la base de las reglas se mantiene haciendo compatibles las mediciones. Se añaden ejemplos y se aclaran definiciones para facilitar el aprendizaje del método.

INTRODUCCIÓN

En el método COSMIC-FFP (de aquí en adelante COSMIC) el tamaño funcional del software bajo medición, se obtiene como un valor proporcional al número de

movimientos de datos de dicho software.

Para ello, el software se entiende que está dividido en distintos **Niveles (Layers)** de forma jerárquica. Cada *Nivel (Layer)* sólo puede comunicarse con sus niveles inmediatamente superior o inferior.

Dentro de cada nivel, el software se agrupa en componentes llamados **Componentes de software (Peers)**, teniendo todos ellos la misma jerarquía.

Los **Procesos Funcionales** se recogen dentro de los distintos *Componentes de software (Peers)*.

El intercambio de información entre distintos niveles se realiza a través de sus respectivos *Procesos Funcionales*.

Un *Proceso Funcional* se forma de distintos **Movimientos de Datos** que son la unidad en la que mide el método.

El **Tamaño de un Proceso Funcional** viene determinado por la suma del tamaño de los movimientos de datos que lo forman.

Las entidades de una aplicación se asocian en **Grupos de Datos**.

El tamaño de una **Pieza de Software**, en un determinado ámbito, se obtiene sumando los tamaños funcionales de los procesos funcionales de los cuales consta dicha pieza de software.

CONCEPTOS

Como decíamos anteriormente, el tamaño funcional del software se obtiene como un valor proporcional al número de movimientos de datos. Por ello, se parte de la Arquitectura del sistema, dividiéndolo en una serie de elementos sobre los cuales describiremos sus interacciones.

El software se divide en distintas capas denominadas **Niveles (Layers)**, siguiendo una jerarquía donde cada nivel sólo se puede comunicar con el nivel inmediatamente superior o inmediatamente inferior.

Un nivel superior confía en la serie de servicios proveídos por los niveles inferiores y un nivel inferior provee de ciertos servicios al nivel superior.

Un nivel intercambia información con otros niveles a través de sus respectivos

Procesos funcionales (ver Ilustración).

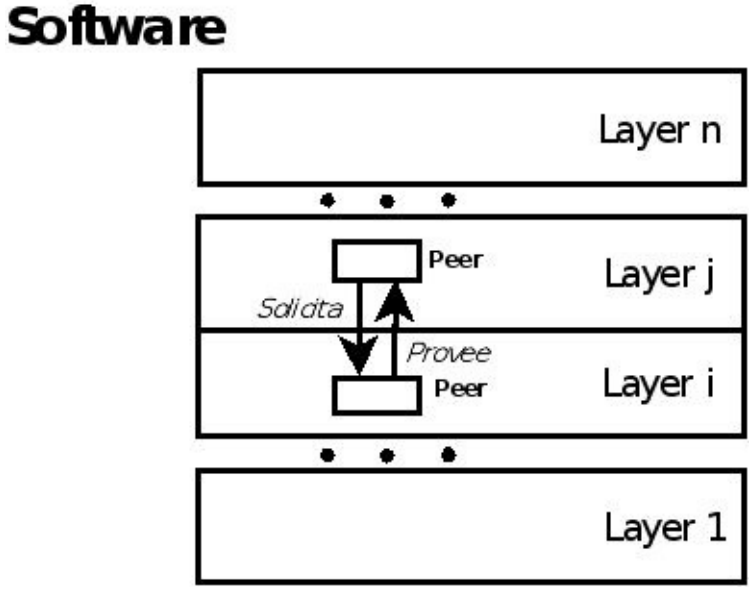


Ilustración - Jerarquía del software según COSMIC

Dentro de un Nivel (Layer), el software se agrupa en componentes de software llamados **Componentes de software** (**Peers)**.

Los Componentes de software (Peers) no están sujetos a ninguna jerarquía entre ellos, todos tienen el mismo valor jerárquico. Los procesos funcionales están dentro de los distintos Componentes de software (Peers) (ver Ilustración).

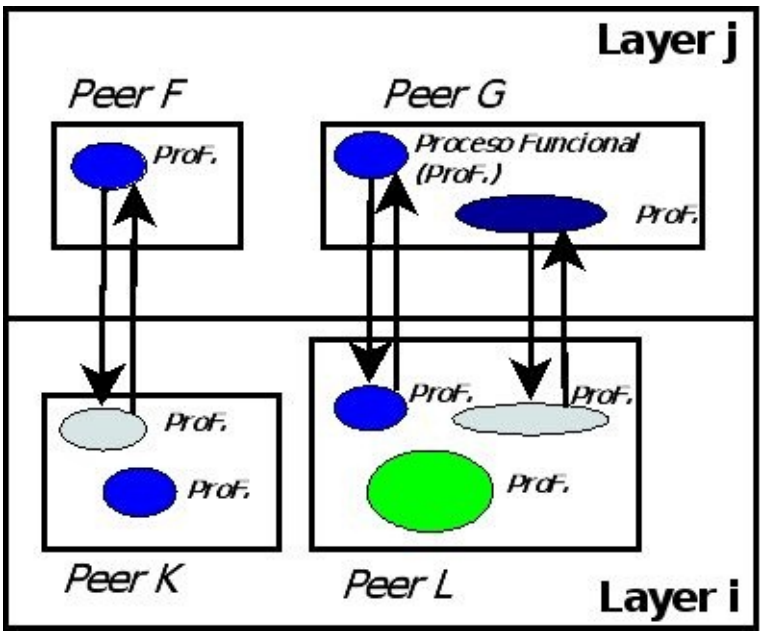


Ilustración - Estructura de los Layers (Niveles) en COSMIC

Los datos que se encuentran en una aplicación software, debemos agruparlos de forma que cada conjunto resultante sea único y distinguible del resto por la colección de atributos que lo forma. Además, dichos conjuntos, deben estar relacionados con un *objeto de interés* recogido dentro de los requisitos funcionales de usuario y les llamaremos **Grupo de Datos (Data Group)**.

Como hemos visto, un software se estructura en Niveles (Layers), los cuales se subdividen en Componentes de software (Peers) y, dentro de éstos, se recogen los Procesos funcionales.

Un **Proceso funcional** se forma de distintos movimientos de datos (Data movements) que son las unidades que mide el método.

Antes de ver los movimientos de datos es necesario conocer otros conceptos.

Para acotar una medición, en el método de *COSMIC*, tenemos que definir nuestro **Propósito** a la hora de realizar la misma y el **Alcance** que tendrá.

Los **Usuarios funcionales** que deberemos tener en cuenta vienen determinados en parte por el propósito de la medición. Cuando el propósito de un conteo es la

determinación del esfuerzo en realizar o modificar una pieza de software, entonces los usuarios funcionales deben ser aquellos para los cuales la nueva funcionalidad o la modificación deban ser provistas.

La **Frontera (boundary)** de la aplicación queda determinada una vez sean identificados los usuarios funcionales del sistema: entre los usuarios y el software.

Además, por definición, entre cada dos Niveles (Layers) queda establecida una frontera y cada uno de los niveles implicados es usuario funcional del otro.

De la misma manera, entre dos componentes de software (peer) queda establecida de igual manera una frontera.

Los **Movimientos de datos** identificados dentro del método *COSMIC* son 4 (ver Ilustración):

- **Entry (E)**. *Entrada*.
  - Mueve un único grupo de datos a través de la frontera de la aplicación hacia el proceso funcional del cual forma parte.
  - Si hay más de un grupo de datos implicado en la entrada de un proceso funcional, deberá contarse una entrada por cada uno de ellos.
  - Una Entrada, ni Lee, ni Escribe información ni tampoco muestra nada en la Salida.

- **Exit (X).** *Salida*.
  - Mueve un único grupo de datos a través de la frontera de la aplicación desde el proceso funcional del cual forma parte hasta uno de los usuarios funcionales.
  - Si hay más de un grupo de datos implicado en la Salida de un proceso funcional, deberá contarse una salida por cada uno de ellos.
  - Una Salida, ni Lee, ni Escribe información ni tampoco recibe un dato de Entrada.

- **Read (R)**. *Lectura.*
    - Mueve un único grupo de datos a través de la frontera de la aplicación desde el medio de almacenamiento persistente hasta el proceso funcional del cual forma parte.
    - Si hay más de un grupo de datos implicado en la Lectura de un proceso funcional, deberá contarse una lectura por cada uno de ellos.
    - Una Lectura, ni Escribe información ni tampoco recibe ni muestra un dato del exterior.

- **Write (X)**. *Escritura*.
    - Mueve un único grupo de datos a través de la frontera de la aplicación desde el proceso funcional del cual forma parte hasta el medio de almacenamiento persistente. Si hay más de un grupo de datos implicado en la escritura de un proceso funcional se deberá contar una Escritura por cada uno de ellos.
    - Una Escritura, ni Lee información ni recibe ni envía un dato de/al exterior.
    - Un requerimiento que exija eliminar un Grupo de Datos se contabilizará como una Escritura.

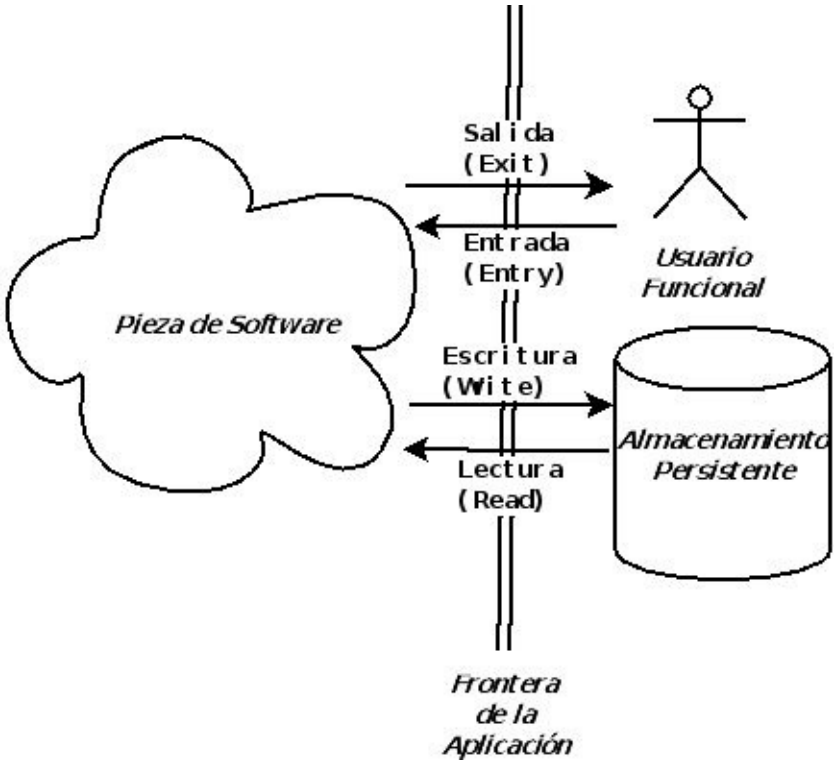


Ilustración - Movimientos de datos (Data movements) en COSMIC

Cualquier procesamiento o manipulación que deban sufrir los datos tratados por cualquiera de los movimientos de datos, se considerará incluido dentro del propio movimiento de datos.

MÉTODO

Consiste en tres fases que se realizan de forma secuencial: **Fase de Estrategia de Medición, Fase de Mapeo** y **Fase de Medición**.

En la **Fase de Estrategia de Medición (***the Measurement Strategy Phase***)** se aplica el Modelo de Contexto del Software al software que va a ser medido. En esta fase, se definen el propósito (*purpose*), el alcance (*scope*), se identifican los usuarios funcionales y el nivel de granularidad requerido en la medición.

En la **Fase de Mapeo** (*the Mapping Phase*) se aplica el Modelo Genérico del Software al software que va a ser medido. En esta fase, se identifican los procesos funcionales, los objetos de interés y los grupos de datos y, opcionalmente, los atributos de datos.

En la **Fase de Medición** (*the Measurement Phase*) se obtienen las mediciones actuales del tamaño. En esta fase, se identifican los movimientos de datos y se suman en función de la fórmula requerida según el propósito.

Para la identificación de los procesos funcionales hay que tener en cuenta que como mínimo deben estar incluidos dentro de un **FUR** (*Functional User Requirement* ó *Requerimiento Funcional de Usuario*).

Todo proceso funcional debe contemplar al menos dos movimientos de datos: una Entrada (Entry) y una Salida (Exit) ó una Escritura (Write), es decir que como mínimo debe aceptar un dato de entrada y debe mostrar algo en la salida o debe guardar en el almacenamiento persistente.

Un movimiento de datos tiene un valor asignado de **1 CFP** (Cosmic Function Point).

El tamaño de un proceso funcional viene determinado por la suma del tamaño de los movimientos de datos que lo forman.

El tamaño de una pieza de software, en un determinado ámbito, se obtiene sumando los tamaños funcionales de los procesos funcionales de los cuales consta dicha pieza de software.

En el caso de que estemos hablando de **modificaciones de software existente**, el tamaño de la modificación de un proceso funcional es la suma de los tamaños de

los movimientos de datos que han sido añadidos, modificados o eliminados de dicho proceso funcional.

De igual forma, el tamaño de los cambios de una pieza de software en un determinado ámbito se obtendrá como la agregación de los tamaños de los cambios de los procesos funcionales de los cuales consta dicha pieza de software. Es decir, sólo se contabiliza aquello que se ha modificado.

La suma total nos dará el tamaño funcional del software que estamos midiendo.

FiSMA

FiSMA es la *Finnish Software Measurement Association* ó *Asociación Finesa de Medición del software*, una organización sin ánimo de lucro cuyo objetivo es alcanzar una gestión eficaz a través de la mejora de la calidad y de la medición del software.

Su sitio web es el siguiente: http://www.fisma.fi/in-english

ORIGEN

FiSMA comenzó en **1992** con el nombre de *LATURI user group*.

En **1998** cambió el nombre por el actual de *FiSMA*, además de expandir sus operaciones iniciales a las actuales.

La versión actual del manual del método es la 1.1.

INTRODUCCIÓN

FiSMA se define como un método parametrizado para medir todo tipo de software.

Reemplaza al método anterior conocido como *FSM method Experience 2.0 Function Point Analysis (FPA)* que fue aplicado ampliamente en Finlandia desde **1997**.

El Método de FiSMA es un método orientado al Servicio y no orientado al proceso, esto significa que en el método FiSMA **se identifican cada uno de los distintos servicios que provee un software y no todos los procesos funcionales soportados** por dicho software.

MÉTODO

Para contabilizar los servicios del software se definen 7 **Clases de**

Componentes Funcionales Base (BFC Classes).

Cada Clase se forma de distintos **Tipos de Componentes Funcionales Base (BFC types)**.

Las 7 Clases de Componentes Funcionales Base son:

- **Servicios de consulta y navegación interactivos para el usuario final (q).** Incluye a datos y/o servicios con entrada y salida de información que no realizan labores de mantenimiento en el sistema.

- **Servicios de Entrada Interactiva para el Usuario final (i).** Incluye a datos y/o servicios interactivos para el usuario final donde se realizan labores de mantenimiento en el sistema.

- **Servicios de Salida No interactiva para el Usuario final (o).** Incluye a datos y/o servicios que cruzan la frontera de la aplicación que no son interactivos y no mantienen un almacén de datos en el sistema.

- **Servicios de interfaz hacia otras aplicaciones (t).** Incluye a datos y/o servicios que cruzan la frontera de la aplicación. Servicios de interfaz hacia otras aplicaciones con todas las transferencias automáticas de información que mueven datos desde el sistema hasta otra aplicación o dispositivo.

- **Servicios de interfaz desde otras aplicaciones (f).** Incluye a datos y/o servicios que cruzan la frontera de la aplicación. Servicios de interfaz desde otras aplicaciones que reciben transferencias automáticas de grupos de datos proveídos por otra aplicación o dispositivo.

- **Servicios de almacenamiento de datos (d).** Grupo o Colección de datos relacionados y auto contenidos en el mundo real, sobre los cuales el usuario requiere que el software provea al menos de un almacenamiento de datos.

- **<u>Servicios algorítmicos y de transformación</u> (a).** Incluye datos y/o servicios realizados por el software para transformar de forma independiente.

Y a continuación podemos ver los tipos de BFC definidos junto con la clase BFC a la que pertenecen:

Servicios de consulta y navegación interactivas para el usuario final (q)	Servicios de Entrada Interactiva para el Usuario final (i)	Servicios de Salida No interactiva para el Usuario final (o)	Servicios de interfaz con otra aplicación (t)
Identificadores de Función (q1)	1 diálogo de entrada funcional (i1)	Formularios de salida (o1)	Mensajes a otras aplicaciones (t1)
Funciones de Log-in y log-out (q2)	2 diálogos de entrada funcional (i2)	Informes (o2)	Registros Batch a otras aplicaciones (t2)
Listas de funciones (q3)	3 diálogos de entrada funcional (i3)	Email y mensajes de texto (o3)	Señales a dispositivos o a otras aplicaciones (t3)
Listas de selección (q4)		Salida por la pantalla del monitor (o4)	
Consultas de datos (q5)			
Generación de indicadores (q6)			
Listas de Navegación (q7)			

Servicios de interfaz con otras aplicaciones (f)	Servicios de almacenamiento de datos (d)	Servicios de algoritmicos y de transformación (a)
Mensajes de otras aplicaciones (f1)	Entidades o clases (d1)	Rutinas de seguridad (a1)
Registros batch de otras aplicaciones (f2)	Otros tipos de registros (d2)	Rutinas de cálculos (a2)
Señales de otros dispositivos u otras aplicaciones (f3)		Rutinas de simulación (a3)
		Rutinas de formateo (a4)
		Rutinas de limpieza de la base de datos (a5)
		Otras rutinas de manipulación (a6)

Los pasos del método son los siguientes:

1. Recopilar la **documentación** y los elementos del desarrollo que describen los requerimientos funcionales de usuario del software.

2. Determinar el **alcance** de la medición.

3. Determinar los **Requisitos Funcionales de Usuario** que están **dentro del alcance** y que por tanto deben ser medidos.

4. Identificar los **BFC** dentro de los requisitos funcionales de

usuario, midiendo tanto los servicios con interfaz de usuario, como los servicios indirectos que sustentan a otros.

5. **Clasificar** los BFC en las **tipologías** correctas.

6. Asignar los **valores numéricos** apropiados utilizando las tablas para los distintos tipos de BFC.

7. Calcular el **tamaño funcional** como la suma de todos los componentes.

Estimación Temprana de NESMA

Además del método que hemos definido anteriormente, NESMA propone dos alternativas para hacer una estimación temprana del software en Puntos Función. Son los métodos Estimado e Indicativo.

MÉTODO ESTIMADO

Este método consta de los siguientes pasos:

- Determinamos todas las Funciones de Datos y Funciones Transaccionales además de los tipos de las mismas: ILF, EIF, EI, EO y EQ.

- Asignamos a cada Función de Datos (ILF y EIF) una complejidad Baja y a cada Función Transaccional (EI, EO y EQ) una complejidad Media.

- Calculamos el conteo Total de Puntos Función No Ajustados, siguiendo las premisas indicadas.

La diferencia con el método original es que **la complejidad de las funciones se establece por defecto** y no se calcula en función de los requisitos de usuario o de la funcionalidad existente.

MÉTODO INDICATIVO

Este conteo se lleva a cabo de la siguiente forma:

- Determinamos el Número de Funciones de Datos (ILF y EIF)

- Calculamos el Total de Puntos Función No Ajustados de la aplicación con la siguiente fórmula:

$$\text{Indicativo} = 35 \cdot \text{Total ILFs} + 15 \cdot \text{Total EIFs}$$

Como se puede ver en los pasos anteriores, **el tamaño funcional de la aplicación se establece únicamente con la identificación de las Funciones de Datos**.

Las Funciones Transaccionales no se identifican sino que se considera, como aproximación, que por cada *Fichero Lógico Interno (ILF)* existen las siguientes funciones transaccionales:

- 3 Entradas (EI): una para la creación, otra para la modificación y otra para el borrado.
- 2 Salidas (EO)
- 1 Consulta (EQ)

Y que por cada Fichero de Interfaz Externo (EIF) existe:

- 1 Salida (EO)
- 1 Consulta (EQ)

VALIDEZ DE LOS MÉTODOS

Para comprobar la validez de los métodos propuestos, NESMA ha realizado una investigación sobre más de 100 proyectos.

La dispersión entre el **método estimado** y el normal es bastante baja siendo mayor cuando los proyectos superan los 1.300 Puntos Función.

En cuando a la **valoración indicativa** con respecto a la estándar, presenta mucha mayor dispersión aunque para proyectos con un tamaño menor a 500 Puntos Función la diferencia es menor.

Esto puede implicar que dependiendo del tipo de proyecto la correlación entre el número de Funciones Transaccionales y el número de Funciones de Datos se deba modificar.

SiFP

SiFPA (Simple Function Point Association) o Asociación de los Puntos Función Simples, se encarga de difundir el conocimiento sobre el método SiFP (Simple Function Points).

Su sitio web es el siguiente: http://www.sifpa.org

ORIGEN

El Método Simple Function Point (SiFP) está siendo desarrollado por DPO desde **2010**.

INTRODUCCIÓN

Para ser compatible con el Método de Puntos Función de IFPUG, el método Simple de Puntos Función parte de los mismos conceptos que el anterior pero añade un concepto nuevo: el **Nivel (Layer)**.

Para ajustarse al estándar ISO de mediciones funcionales (ISO/IEC 14143-2:2011) no se utiliza el VAF (Value Adjustment Factor) sino que el valor final se correspondería con *Puntos Función No Ajustados*.

Con el método se busca como principales objetivos que permita:

- Un método más simple de aprender
- Un método más simple de aplicar
- Menos discrepancias con los proveedores
- Compatible con el método de IFPUG
- Compatible con el estándar ISO de Medición Funcionales

MÉTODO

El concepto de **Nivel (Layer)** se introduce para poder modelar sistemas que diseñan y utilizan componentes de software reutilizables ya sean técnicos y/o de negocio.

Por ello, las Aplicaciones que identificábamos en IFPUG ahora son conocidas como **MSA (Measurable Software Application o Aplicación de Software**

Medible) y puede representar a un único Nivel (Layer) del Sistema final.

Es decir, podemos tener un MSA que se corresponda con el nivel de Interacción de usuario de nuestro sistema y otro MSA con la capa de negocio, si ambas se forman por componentes reutilizables.

Cada **Función** impactada sólo se cuenta una única vez aunque se utilice en varias aplicaciones, por el hecho de la reutilización.

El Método Simple de Puntos Función, como nos indica su nombre, simplifica al método de IFPUG y lo hace a la hora de identificar las funciones tanto transaccionales como de datos.

Ahora sólo tenemos dos tipos de funciones:

- **Proceso Elemental Genérico Sin Especificar (Unspecified Generic Elementary Process o UGEP).**
    - Englobaría, sin ninguna distinción, las funciones de:
        - Entrada (EI)
        - Salida (EO)
        - Consulta (EQ)
    - Tiene un peso de 4,6 SiFP. Es decir el resultado total será el número de funciones por este valor.

- **Grupo de Datos Genérico Sin Especificar (Unspecified Generic Data Group ó UGDG).**
    - Englobaría las funciones de datos:
        - Ficheros Lógicos Internos (ILF)
        - Ficheros de Interfaz Externo (EIF)
    - Tiene un peso de 7,0 SiFP. Es decir el resultado total será el número de funciones por este valor.

Una vez realizada la identificación de las funciones transaccionales y de las funciones de datos calcularíamos el tamaño en SiFP con la siguiente fórmula:

$$\mathbf{SiFP} = 4{,}6 \cdot \sum(\mathrm{UGEP}) + 7{,}0 \cdot \sum(\mathrm{UGDG})$$

Puntos Función 3D

Este método fue desarrollado por Boeing. Se desconoce si se ha utilizado fuera de la compañía o si se continúa utilizando actualmente dentro de ella.

ORIGEN

El método de Medición en Puntos Función 3D (PF3D) fue desarrollado por *Scott A. Whitmire* entre **1989** y **1992** para la compañía Boeing con el objetivo de poder medir aplicaciones de negocio y sistemas en tiempo real.

MÉTODO

Necesita disponer de mayor cantidad de información del sistema que cualquier otro método. Sobre todo necesita información relativa a la complejidad de los algoritmos, hecho que es su principal inconveniente.

La medición del sistema se realiza teniendo en cuenta no sólo los datos y los procesos del mismo, sino teniendo en cuenta también el flujo de control.

La medición se realiza con respecto a tres componentes (de ahí su nombre):

- **La dimensión de datos.** La misma que en Puntos Función.
- **La dimensión funcional.** Que contabiliza las operaciones necesarias para convertir los datos en salidas, las transformaciones de los datos.
- **La dimensión de control.** Que contabiliza el número de transacciones entre diferentes estados del sistema.

Una medición en Puntos Función 3D tendría la siguiente distribución:

Elemento de Medición	Pesos de complejidad			Subtotal
	Elemento de Medición	Pesos de complejidad	Subtotal	
Estructuras Internas de Datos	$n_b \cdot 7 +$	$n_m \cdot 10 +$	$n_a \cdot 15 =$	$st1$
Datos Externos	$m_b \cdot 5 +$	$m_m \cdot 7 +$	$m_a \cdot 10 =$	$st2$
Número de Entradas de Usuario	$o_b \cdot 3 +$	$o_m \cdot 4 +$	$o_a \cdot 6 =$	$st3$
Número de Salidas de Usuario	$p_b \cdot 4 +$	$p_m \cdot 5 +$	$p_a \cdot 7 =$	$st4$
Número de Peticiones de Usuario	$q_b \cdot 3 +$	$q_m \cdot 4 +$	$q_a \cdot 6 =$	$st5$
Transformaciones	$r_b \cdot 7 +$	$r_m \cdot 10 +$	$r_a \cdot 15 =$	$st6$
Transiciones	$s_b +$	$s_m +$	$s_a =$	$st7$
		Total de Puntos Función 3D		*tpf3d*

Donde:

- **Estructuras internas de datos.** Se refiere a estructuras de datos relativas al software: punteros, listas enlazadas, pilas, colas, etc.

- **Datos externos.** Serían las funciones de datos tal y como se definen en Puntos Función (Archivos e Interfaces).

- **Entradas de Usuario.** Serían las Entradas de Datos (EI) tal como se definen en Puntos Función.

- **Salidas de Usuario.** Serían las Salidas de Datos (EO) tal como se definen en Puntos Función.

- **Peticiones de Usuario.** Serían las Consultas de Datos (EQ) tal como se definen en Puntos Función.

- **Transformaciones.** Son las operaciones internas necesarias para que con la entrada recibida se obtenga la salida deseada.

Por ejemplo:
Multiplicar dos matrices es una operación, leer datos de un archivo y guardarlo en memoria no es una acción (sería una entrada).

- **Transiciones.** Cuando, como resultado de algún suceso el software pasa de un estado a otro.

Por ejemplo:
Un sistema en estado de espera puede pasar al estado Alta cuando se solicita dicha opción en el menú principal.

Para cada una de estas medidas se deben contar cuantas hay de complejidad baja, media y alta.

Luego se procede a multiplicar la frecuencia de cada caso por la complejidad asignada en función del tipo de componente.

Con la suma total obtenemos los puntos función 3D del Proyecto.

En un principio, el resultado del sistema en PF3D se obtenía sumando los resultados obtenidos en cada una de las tres componentes pero luego el sistema evolucionó para mantener el valor de cada componente por separado.

¿Qué Método de Puntos Función Utilizar?

Muchos son los puntos de vista desde los que podemos pretender encontrar el mejor método de medición en Puntos Función.

Podemos necesitar encontrar el método que mejor se ajuste a nuestro tipo de aplicación: tiempo real, sistema comunicaciones, de gestión,…

Podemos necesitar compararnos con el resto de la industria para poder ver si nuestro rendimiento está acorde con lo que realizan otros o si el rendimiento del proveedor se ajusta a la generalidad.

Podemos necesitar una herramienta que nos permita estimar y predecir con antelación para que minimicemos los desajustes en presupuesto y recursos en el desarrollo.

Todo ello, nos lleva a tomar una serie de decisiones de consenso para cada caso particular, para cada empresa o para ciertas líneas de cada empresa.

A continuación vamos a exponer las razones por las cuales el método de IFPUG es el que se ajusta, en general, a un rango más amplio de las necesidades de la industria, aunque esto no quita que para ciertos casos sea mejor utilizar otros métodos:

1. **Primer método desarrollado de medición en Puntos Función.**

   Al ser el primer método desarrollado de medición en puntos función lleva muchos años bajo estudio y refinamiento. Es un método bastante estable, aceptado (sobre todo la parte no ajustada) y nos permite basar nuestras mediciones en un pilar sólido.

2. **El resto de métodos parten de sus conceptos.**

   El resto de métodos de medición en puntos función, en general, son ideas que parten del refinamiento del desarrollo de los puntos función.

   Todos ellos se comparan con este método a la hora de explicar sus nuevas alternativas e incluso, en muchos casos, proveen de herramientas de conversión de puntos función IFPUG a puntos función de sus métodos.

Esto nos garantiza que en un futuro si necesitáramos optar por uno de los otros métodos, no habrá que tirar nuestras mediciones sino que podríamos adaptarlas y seguir contando con una base de datos histórica.

3. **Es el estándar de facto.**

Es el método más extendido actualmente de mediciones en puntos función. Existe gran cantidad de información sobre él, ejemplos, Web dedicadas a su soporte, intercambio de información, foros, etc.

4. **Mayor facilidad para compararnos con otros.**

Existen muchas bases de datos de valores históricos de muchas compañías con las cuales poder compararse y ver la evolución frente al mercado, ajustar nuestras productividades con las del resto de la industria o poder negociar los contratos con los proveedores con medidas del mercado.

5. **Sencillez del método.**

El método es sencillo de aplicar, incluso más que algunos de sus competidores.

6. **Necesidad de poca información para realizar estimaciones.**

Una gran ventaja con respecto al resto de métodos es que necesita mucha menos información a la hora de realizar estimaciones.

Este hecho en otros métodos es un hándicap ya que para poder realizar una estimación debemos tener la misma información que al final del diseño funcional, perdiendo una buena oportunidad de realizar una estimación en la fase de toma de requerimientos o plan de proyecto.

7. **Se ajusta a un alto porcentaje de aplicaciones de la industria.**

Su mayor hándicap es la medición de sistemas en tiempo real y los sistemas con alto procesamiento lógico.

En este punto el resto de métodos se encuentra en la misma situación

frente a los sistemas de alto procesamiento lógico y superándolo sólo algunos que pueden medir sistemas en tiempo real y de alto procesamiento lógico.

El general resulta un balance aceptable.

APLICACIONES EN TIEMPO REAL Y ALTO PROCESAMIENTO

Para este tipo de aplicaciones o para aplicaciones que realicen mucho cálculo o procesamiento, por ejemplo aplicaciones científicas, debemos optar por otro método de medición en puntos función que nos permita recoger todas estas acciones.

Siguiendo los mismos criterios que hemos utilizado para recomendar IFPUG de forma general, en este caso, el método a recomendar sería COSMIC FFP. Las razones son las mismas: es el segundo método más utilizado en la industria, ha sido diseñado para medir aplicaciones en tiempo real y podemos compararnos en varias bases de datos, no tantas como con IFPUG pero existen.

Una mención destacable es el método FiSMA que por basarse en un concepto diferente al resto y sencillo de implementar nos permite recoger todos los servicios del software de forma fácil. Como hándicaps son su poca expansión: sólo se utiliza en Finlandia y la necesidad de información: mayor que en IFPUG aunque similar a COSMIC.

Por otro lado, si necesitamos velocidad en la medición podemos optar por alguna de las opciones rápidas como la estimación temprana de NESMA o el método de SiFP.

Al final la decisión está en tus manos.

Capítulo 4 Otros Métodos de Medición Funcional

Puntos Característica (Feature Points)

Su desarrollo se comenzó partiendo de los Puntos Función, cambiando el nombre del método para evitar equívocos.

ORIGEN

Este método fue propuesto por *Caper Jones* entre **1986** y **1987**. Su propósito era extender el método de puntos función para que pudieran realizarse mediciones de algoritmos científicos y de ingeniería.

INTRODUCCIÓN

Se ha utilizado para medir distintos tipos de software como sistemas en tiempo real, sistemas embebidos, aplicaciones CAD, software para inteligencia artificial, etc.

Estos tipos de sistemas, se diferencian principalmente de los sistemas de gestión en la complejidad de los algoritmos que implementan (cuestión que no es medible con el método de puntos de función) y, por otra parte, cuentan con pocas interacciones con el usuario (pocas entradas y salidas).

Según algunos estudios, en los tipos de sistemas descritos, un análisis en Puntos Función suele medir entre un 20% o un 30 % menos de puntos función de los que contaría si se utilizara un modelo que tuviera en cuanta la complejidad algorítmica de los mismos.

Hay estudios que presentan resultados de ratios de comparación entre las mediciones que se obtendrían en Puntos Función y en Puntos Característica para determinados tipos de sistemas (ver Tabla):

Tipo de sistemas	Ratios	
	Puntos Función	Puntos Característic
Orientados a la Gestión (con proceso por	1.00	0.80
Orientados a la Gestión (interactivos)	1.00	1.00
Base de Datos On-line	1.00	1.00
Sistemas para el Control de Procesos	1.00	1.28
Embebidos / En tiempo real	1.00	1.35
Sistemas para automatización industrial	1.00	1.50

Tabla - Ratios de comparación por Sistemas entre Puntos Función y Puntos Característica

De la anterior tabla se pueden obtener interpretaciones interesantes, como por ejemplo que para sistemas orientados a la gestión con proceso por lotes mediremos un 20% menos puntos característica que puntos función.

También que para sistemas de automatización industrial mediremos un 50% más de puntos característica que de puntos función.

MÉTODO

Los puntos característica parten con los mismos elementos que los Puntos Función eliminando la complejidad de los mismos (se presuponen todos de complejidad media) y añadiendo además el número de algoritmos como una variable más.

Los **algoritmos** se definen en este método como *un problema computacional de alcance y límites bien definidos, que se implementa dentro de una determinada aplicación informática.*

La **complejidad de los algoritmos** se mide entre 1 y 10. Cuando son de poca complejidad y se basan en operaciones aritméticas básicas se le asigna el valor mínimo (1) y cuando se basan en complicadas ecuaciones matemáticas, operaciones con matrices, etc. se aplica el valor máximo (10).

Para evitar la subjetividad a la hora de medir dicha complejidad, debemos fundamentarla en el número de pasos de cálculo o reglas de las que consta el algoritmo y en el número de factores o datos que necesita dicho algoritmo para realizar su función. Así podemos tomar una decisión objetiva de su complejidad.

Se tienen también una serie de reglas a la hora de realizar el conteo de los

algoritmos objeto de estudio:

- Los algoritmos deben estar definidos para problemas resolubles, con alcance y límites bien definidos.
- Deben tener una duración limitada en el tiempo de su ejecución. Deben concluir en un tiempo finito.
- Deben ser precisos y sin ambigüedades.
- Deben admitir una entrada (o valores de comienzo) y retornarán una salida (o valores resultado).
- Un algoritmo puede estar compuesto de uno o varios subalgoritmos o bien puede llamar a otros algoritmos para completar la función que realice.
- Todos los pasos de los que conste un algoritmo deben poderse implementar en la máquina destino haciendo uso de las sentencias típicas de la programación estructurada: if-then-else, do-while, case, etc.

Para el cálculo veamos la siguiente tabla (ver Tabla):

Parámetro significativo	Número	Factor de ponderación	Totales
Número de Algoritmos	*n*	X 3 =	st1
Número de Entradas (EI)	*o*	X 4 =	st2
Número de Salidas (EO)	*p*	X 5 =	st3
Número de Consultas (EQ)	*q*	X 4 =	st4
Número de Ficheros de Datos (ILF)	*r*	X 7 =	st5
Número de Interfaces (EIF)	*s*	X 7 =	st6
Total Puntos Caracteristica Sin Ajustar			uafp
Ajuste de Complejidad			ca
Total de Puntos Caracteristica Ajustados			afp

Tabla - Plantilla de Medición de Puntos Característica (Feature Points)

Donde vemos que para cada uno de los parámetros significativos se realiza la contabilización de cuantos hay y se multiplican por su factor de ponderación, para así calcular su contribución en puntos característica.

Después de realizarlo, se suman todos los subtotales obtenidos y se obtiene el

valor sin ajustar. Una vez multiplicado por el ajuste de complejidad del proyecto, nos permite obtener el valor de los puntos característica ajustados para el objeto de estudio.

Con todo esto vemos que los Puntos Característica son un superconjunto de los Puntos Función. Aún así, han tenido poco calado y hay pocos proyectos que se hayan medido utilizando este método.

Puntos Objeto

Su objetivo era ayudar en la estimación de sistemas desarrollados con tecnologías 4GL ó similares, apoyados por herramientas CASE.

Su estudio se realizó sobre aplicaciones *ICE* (*Integrated CASE Environnement*) y se tuvo siempre en cuenta para que fuera utilizado sobre herramientas CASE y repositorios de éstas mismas.

Actualmente no se utiliza por sí mismo como método de estimación sino que se suele utilizar como una entrada alternativa, mejor que las Líneas de código, para el modelo de estimación de costes, *COCOMO*.

ORIGEN

Desarrollado al comienzo de los 90 por *Rajiv D. Banker, Robert J. Kauffman* y *Rachna Kumar*, fue presentado por primera vez en un trabajo de **1991**.

INTRODUCCIÓN

Una de las principales ventajas, que presentaba este método en el estudio presentado en 1991, es que el esfuerzo para obtener la estimación del tamaño funcional del sistema era, en promedio, cercano a la mitad del necesario para realizar la misma estimación bajo el método de Puntos de Función.

Este método se basa en medir los objetos de una aplicación pero estos objetos no se refieren a los mismos que se describen en la programación orienta a objetos sino a *objetos de negocio* u *objetos de interés para el negocio*.

Los **objetos de negocio** que se tienen en cuenta son:

- **Pantallas** (*Screens Definitions*). Entendidas como la representación lógica de la imagen en pantalla y trasmite la funcionalidad que

normalmente está asociada con el interfaz de usuario.

- **Informes** (*User Reports*). La información que se muestra al usuario con distintas secciones de valores mantenidos/generados por el sistema.
- **Módulos 3GL**. Este objeto de negocio se refiere a módulos ó componentes reusables desarrollados en un lenguaje de Tercera Generación para el sistema.

MÉTODO

El concepto de reusabilidad se tuvo siempre en cuenta en el desarrollo de este método de estimación y, por ello, se añade el componente **Módulo 3GL**, que puede ser utilizado por la aplicación o que debe ser utilizado por otros aunque no construido.

Este concepto también está heredado del punto de partida del método, el diseño de aplicaciones con herramientas CASE e intenta aprovechar los repositorios con la información relativa a los componentes de las distintas aplicaciones que se pueden utilizar pero se construyen únicamente una vez.

El método consta de los siguientes pasos:

1) Determinar todos los objetos de los que está compuesto el sistema. A cada uno de ellos se le asigna un peso numérico en función de la complejidad que se calcule.

Hay tres niveles de complejidad: Simple, Media y Compleja.
Para las pantallas y los informes, la complejidad se determina en función del número de consultas y tablas de datos accedidas.

Para cada módulo 3GL la complejidad siempre es la misma y el valor en puntos objeto también: 10.

2) El valor en Puntos Objeto del sistema es igual a la suma del valor de Puntos Objeto de todos y cada uno de sus componentes (OP).

3) El valor de los Nuevos Puntos Objeto que aporta el sistema será igual al valor anterior restándole los Puntos Objetos que se reutilizan. Para ello se calcula el porcentaje de reutilización.

$$NOP = OP \cdot (1 - \%\text{Reutilización})$$

4) Con el paso anterior ya tendríamos los Puntos Objeto de nuestro sistema pero el método continúa hasta el cálculo del esfuerzo en el desarrollo. Para ello se tiene en cuenta la productividad del entorno y la productividad del programador.

El valor nominal de la **productividad del entorno** es 1 y se ajusta en función de la diferencia con respecto a éste de nuestro entorno.

El valor nominal de la **productividad de los programadores** es de 13 NOPs (Nuevos Puntos Objetos) por mes y está siempre en un rango de 4 a 50 NOPs. Con ello el cálculo del esfuerzo nos quedaría como:

$$Esfuerzo = \frac{(NOP \cdot \text{Productividad Entorno})}{\text{Productividad Programadores}}$$

Para determinar la complejidad de los objetos de negocio del sistema debemos recurrir a las siguientes tablas.

La complejidad de las pantallas se calcula en función de las consultas y a las

tablas de datos a las que accede (ver Tabla).

Número de Consultas	Número de Tablas de Datos		
	<4	de 4 a 7	>7
<3	Simple	Simple	Media
de 3 a 7	Simple	Media	Compleja
>7	Media	Compleja	Compleja

Tabla - Cálculo de la complejidad de las Pantallas en Puntos Objeto

La Complejidad de los informes se determina en función de las secciones definidas dentro del mismo y de los accesos a tablas de datos (ver Tabla):

Número de Secciones	Número de Tablas de Datos		
	<4	de 4 a 7	>7
<2	Simple	Simple	Media
de 2 a 3	Simple	Media	Compleja
>3	Media	Compleja	Compleja

Tabla - Cálculo de la complejidad de los Informes en Puntos Objeto

Una vez determinada la complejidad, el valor en Puntos Objeto de cada una de ellas en función del tipo de objeto de negocio se obtiene de la siguiente tabla (ver Tabla):

Tipo de Objeto	Simple	Media	Compleja
Pantalla	1	2	3
Informe	2	5	8
Módulo 3GL	10		

Tabla - Pesos en Puntos Objeto de cada uno de los Tipos de Objeto en función de la complejidad

Puntos Casos de Uso

Este método ha sido ampliamente utilizado por la empresa Rational. Su principal ventaja es su rápida adaptación a empresas que ya estén utilizando la técnica de Casos de Uso.

ORIGEN

El método de Puntos Casos de Uso (Use Case Points) fue desarrollado en **1993** por *Gustav Kamer*, bajo la supervisión de *Ivar Jacobson* (creador de los casos de uso y gran promotor del desarrollo de UML y el Proceso Unificado).

MÉTODO

Para el cálculo se procede de forma similar a Puntos Función: se calcula una **cuenta no ajustada Puntos Casos de Uso (UAUCP)**, asignando una complejidad a los actores y a los casos de uso.

Esta complejidad será ponderada con un **Factor de Ajuste técnico** y por un **Factor de Ajuste relativo al entorno de implantación**, obteniendo tras ello una cuenta de Puntos Casos de Uso Ajustados.

Veamos a continuación en detalle los pasos del método:

Paso 1. Clasificar cada interacción entre actor y caso de uso según su complejidad y asignar un peso en función de ésta.

Para poder clasificar la complejidad de los actores debemos analizar la interacción de éste con el sistema que se va a desarrollar.

La complejidad de los actores puede corresponderse con una de las tres categorías posibles:

a. *Simple*. Representa a otro sistema con una API definida. Se le asigna un peso de valor 1.

b. *Media*. Representa a otro sistema que interactúa a través de un protocolo de comunicaciones. TCP/IP o a través de un interfaz por línea de comandos. Se le asigna un peso de valor 2. Por ejemplo:

TCP/IP o a través de un interfaz por línea de comandos.

c. *Compleja*. La interacción se realiza a través de una interfaz gráfica. Se le asigna un peso de valor 3.

Tipo de Interacción	Peso Asignado
Simple (a través de API)	**1**
Media (a través de protocolo)	**2**
Compleja (a través de interfaz gráfica)	**3**

Paso 2. Calcular la complejidad de cada caso de uso según el número de transacciones o pasos del mismo.

Para calcular la complejidad de un caso de uso debemos determinar el número de transacciones, incluyendo los caminos alternativos.

Se entiende por **transacción** a un conjunto de actividades atómicas, donde se ejecutan todas ellas o ninguna.

En función del número de transacciones que posee un caso de uso se clasifica el caso de uso como simple, medio o complejo, siendo la asignación de pesos la que se muestra en la tabla siguiente:

Número de transacciones del caso de	Tipo	Peso
menor o igual que 3	Simple	5
mayor o igual que 4 y menor que 7	Medio	10
mayor o igual que 7	Complejo	15

Paso 3. Calcular los Puntos Casos de Uso No Ajustados (UUCP) del sistema.

Se obtienen sumando los Puntos Casos de Uso de todos y cada uno de los actores y casos de uso que se han identificado y catalogado en función de su complejidad.

Paso 4. Cálculo de los Factores Técnicos (TCF).

A cada uno de los Factores Técnicos de la tabla siguiente se le asigna un valor de influencia en el proyecto entre 0 (no tiene influencia) a 5 (esencial), 3 se considera de influencia media.

Factor	Descripción	Peso	Influencia
R1	Sistema Distribuido	2	n
R2	Objetivos de rendimiento	1	o
R3	Eficiencia respecto al usuario final	1	p
R4	Procesamiento complejo	1	q
R5	Código reutilizable	1	r
R6	Instalación sencilla	0,5	s
R7	Fácil utilización	0,5	t
R8	Portabilidad	2	u
R9	Fácil de cambiar	1	v
R10	Uso concurrente	1	w
R11	Características de seguridad	1	x
R12	Accesible por terceros	1	y
R13	Se requiere formación especial	1	z

Obtenidos los grados de influencia se multiplican por el peso de cada factor y con la siguiente fórmula se calcula el Factor Técnico que aplica:

$$\mathbf{TCF} = 0.6 + (0.01 \cdot \sum_{i=1}^{13} R_i)$$

Paso 5. Cálculo de los Factores de Entorno.

A cada uno de los Factores de Entorno de la tabla siguiente se le asigna un valor de influencia en el proyecto entre 0 (no tiene influencia) a 5 (esencial), 3 se considera de influencia media.

Factor	Descripción	Peso	Influencia
R1	Familiar con RUP	1,5	n1
R2	Experiencia en la aplicación	0,5	n2
R3	Experiencia en orientación a objetos	1,0	n3
R4	Capacidades de análisis	0,5	n4
R5	Motivación	1,0	n5
R6	Requisitos estables	2,0	n6
R7	Trabajadores a tiempo parcial	-1,0	n7
R8	Lenguaje complejo	-1,0	n8

Obtenidos los grados de influencia se multiplican por el peso de cada factor y con la siguiente fórmula se calcula el Factor de Entorno que aplica:

$$\mathbf{EF} = 1.4 - 0.03 \cdot \sum_{i=1}^{8} R_i$$

Paso 6. Obtención de los Puntos Casos de Uso Ajustados.

Una vez tenemos los dos factores, calculamos el valor ajustado de Puntos Casos de Uso con la siguiente fórmula:

$$\mathbf{UCP} = UAUCP \cdot TCF \cdot EF$$

Obtenido el número de Puntos Casos de Uso y si queremos obtener el esfuerzo necesario para llevarlos a cabo, en el método provee de un factor de productividad.
El autor propone un valor de *20 horas/persona* aunque existen distintas

propuestas sobre el mismo.

Este esfuerzo calculado no abarcaría a todas las fases del proyecto sino únicamente a la codificación de los Casos de Uso, no estando contempladas otras fases del desarrollo.

Por tanto, para calcular el esfuerzo total del proyecto habría que estimar el esfuerzo en realizar el resto de actividades del proyecto y sumarlas a las obtenidas por el método de Puntos Casos de Uso.

Capítulo 5 Métodos de Medición No Funcional

Lógica Difusa (Fuzzy Logic)

Es una forma de estimación que se basa en la clasificación de las características del software en distintos tipos:

- Muy Pequeño
- Pequeño
- Medio
- Grande
- Muy Grande

Para poder **conocer cuantas líneas de código categorizan** a una característica como *Muy Pequeña, Pequeña, Media,…* tenemos que contar con un **histórico** (una base de datos histórica) donde se hayan almacenado estos valores y podamos obtener el valor medio para cada una de las categorías.

Es muy importante contar con un histórico propio ya que los valores de las líneas de código por categoría podrían cambiar entre distintas empresas/compañías.

Para obtener **el tamaño total del software** sólo debemos contar cuantas características están en cada categoría, multiplicarlas por el valor promedio de líneas de código de las mismas y sumarlas todas.

Esto nos dará el valor estimado en líneas de código de esa aplicación.

La validez del método se basa en el número de características identificadas, es decir cuanto mayor sea el número de características que hemos identificado más probable será que la suma de las líneas de código de todas ellas se acerque al promedio y por tanto nos desviemos poco en la estimación.

Catálogo de Componentes (Standard Components)

Si estamos en un entorno donde el desarrollo del software sigue una misma arquitectura o una arquitectura muy parecida, podemos intentar estimar el tamaño de los elementos estableciendo un catálogo de componentes (*standard components*).

El primer paso pues, es identificar que es un **componente estándar** para nosotros. Esto dependerá de nuestros sistemas, de lo que hayamos desarrollado hasta el momento. Podemos definir:

- Ventanas
- Ficheros
- Tablas de la base de datos
- Widgets
- Programas Cobol
- …

Todo depende de nuestro caso particular.

Una vez identificado, el segundo paso es proceder como en el método de *Lógica Difusa (Fuzzy Logic)*, calculamos el promedio de líneas de código de nuestro sistema para cada componente.

Puntos de Historia

Las metodologías ágiles han cambiado muchos conceptos en el desarrollo del software, entre ellos la forma de realizar estimaciones.

El **método de puntos de historia** se desarrolló no para obtener un valor exacto en horas, de esfuerzo, para el desarrollo de una historia de usuario (cada una de las partes en que se divide la funcionalidad a desarrollar en las metodologías ágiles) sino **como una manera de dimensionar y relacionar la complejidad de las historias de usuario con respecto a otras**.

MÉTODO

Como primer paso del proceso se selecciona una **historia de usuario** y se le asigna **una complejidad nominal** que servirá de referencia para catalogar el resto de historias de usuario:

Como esta historia es de complejidad X esta otra, que en comparación es menos compleja, será de complejidad i, donde i es menor que X

o

como esta historia es más compleja será de complejidad j, donde j es mayor que X.

Los valores que se utilizan para representar la complejidad **no tienen una catalogación absoluta** sino que su valor es relativa a su posición en la escala.

Se comenzó utilizando la serie de Fibonacci: *1,2,3,5,8,13,21, …,* aunque para evitar que se pensara que había una precisión matemática en los valores a partir de cierto número se sustituyeron por otros aproximados: *3,5,8,13,40,100,…*

Se recomienda no utilizar ni el 1 y ni el 2 porque no implican mucha diferencia con respecto al 3.

También pueden utilizarse los siguientes valores: *Extra Small, Small, Medium, Large, Extra Large*.

E incluso se ha utilizado la potencia de dos: 1, 2, 4, 8, 16, 32, etc.

Como el método se basa en la comparación de historias de usuario ya realizadas se necesita contar con una línea base de historias completadas por el equipo.

Que el valor sea relativo y no absoluto significa que si a una historia le asignamos el 8 y a otra el 100 lo único que podemos decir es que la de valor 100 es muchísimo más compleja que la de valor 8 pero no tenemos ningún multiplicador que aplicado al valor nos dé el número de horas que costará desarrollar las historias.

La complejidad de las historias, los puntos de historia, **no se pueden comparar a horas de esfuerzo** ya que el sentido que tienen es catalogar la dificultad de la tarea. El número de horas que nos lleve realizarlas dependerá de la capacitación y/o capacidad de la persona que la lleve a cabo, la carga de trabajo del equipo, etc. y, por ello, dicho valor variará dependiendo de la situación.

MEDIDA PROPIA DE UN SÓLO EQUIPO

El **principal problema** que supone la utilización de los puntos de historia es que **son relativos a cada equipo de desarrollo** y, por ello, **no podemos comparar los puntos de historia medidos por un equipo con los de otros equipos,** ya que la utilización de los valores puede ser diferente.

Es más, tampoco podemos hacer comparaciones de la velocidad de desarrollo de cada uno de los equipos por el número de puntos de historia que hayan implementado ya que podemos estar comparando naranjas con manzanas.

PLANIFICACIÓN ÁGIL

En una planificación siguiendo metodología ágil, se seleccionan cuantos puntos de historia se podrán entregar en la siguiente iteración, es decir cuántas historias de usuario se podrán implementar en función de la productividad del equipo, de la carga de trabajo y del resto de factores ambientales. Con ello, las tareas de planificación se simplifican y se logran valores bastantes más cercanos a la realidad.

T-SHIRT Sizing

En el caso del T-Shirt Sizing se vuelve a anteponer el valor relativo con respecto a la precisión. Suele aplicarse en entornos donde el cliente no tiene un conocimiento técnico amplio y, por tanto, sólo desea conocer la complejidad relativa de las tareas a realizar, con objeto de poder mejorar el control y la planificación de éstas.

Las tareas se catalogan en los siguientes valores: Small, Medium, Large, eXtra Large,… de forma similar a las tallas de las camisetas de ahí el nombre del método (S, M, L, XL, XXL, XXXL).

En una de las variantes del método, desde el negocio se catalogan las tareas/características a implementar con los valores indicados siguiendo un criterio de mayor o menor importancia para ellos. Por otro lado, los desarrolladores catalogan las mismas desde la perspectiva del coste del desarrollo.

Así, obtenemos una lista con los dos valores para cada una de las características. Esta lista permite al cliente seleccionar y priorizar las tareas en función de su importancia desde un punto de vista de negocio y su esfuerzo desde un punto de vista de desarrollo.

La relación entre el esfuerzo en el negocio y el esfuerzo en el desarrollo se puede cuantificar a través de una tabla que asigne valores numéricos a cada par de valores.

Con ella completaríamos los dos primeros valores calculados y junto a ellos nos ayudaría a tomar la decisión de planificación y priorización.

SNAP

Es la respuesta de IFPUG al gran hándicap del método de Puntos Función aplicado a la estimación de esfuerzos: ¿cómo medimos los requerimientos no funcionales de usuario?

La respuesta ha tardado en llegar pero se llama Software Non-functional Assessment Process (SNAP).

Todavía está en desarrollo y no está completamente definido pero ya es un avance importante. Su objetivo es dimensionar todos los requerimientos no funcionales que solicita el usuario y que quedaban excluidos del tamaño funcional del software en una medición en puntos función, a través de unas categorías y subcategorías.

INTRODUCCIÓN

SNAP se encarga de mapear cada uno de los requisitos no funcionales de usuario en una subcategoría.

El método SNAP mide o estima un proyecto o aplicación en los llamados Puntos SNAP ó SNAP Points. Estos puntos son el complemento de los puntos función y dan una medida del tamaño no funcional.

Para medir los Puntos SNAP de las subcategorías, éstas tienen asociada una unidad de conteo.

En función del número de unidades de conteo impactadas y de la complejidad de las mismas obtendremos una medición mayor o menor.

CONCEPTOS

En SNAP se introducen una serie de conceptos que son nuevos pero, a la vez, se modifican ligeramente algunos conceptos con respecto al método de IFPUG y es importante tener claras esas modificaciones ya que hacen variar las mediciones.

Una **categoría** es grupo de procesos, actividades o componentes que son

utilizados para cumplir un requisito no funcional.

Una **subcategoría** es un componente, proceso o actividad ejecutado dentro de un proyecto para cumplir con un requisito no funcional. Lo que se mide son los componentes, procesos o actividades que son realizados para cumplir los requisitos no funcionales de usuario, no éstos tal cual.

Un Requisito No funcional del Proyecto es optimizar la consulta C1 para que tarde 1 segundo en lugar de los 7 segundos actuales. Para ello, se crea una Vista en la Base de Datos que mejora el tiempo de ejecución.

Lo que se mide con SNAP es la creación de la Vista en la Base de Datos a través de la Subcategoría Tecnología de Base de Datos.

Las **SNAP Counting Units** o SCU son las unidades en las cuales se miden las subcategorías, es decir el requisito no funcional se asociará a una subcategoría y dentro de ésta se medirán las unidades de conteo SNAP impactadas.

A todas las SCU impactadas se les calculará su complejidad.

Finalmente sumaremos todos los valores calculados para obtener el resultado de puntos SNAP totales.

Las SCU son, en la mayoría de casos, un Proceso Elemental. Son un proceso elemental tal como se define en IFPUG de ahí la relación que comentábamos al principio que tiene con este método en su propia definición.

Los **Datos de codificación** o **Code Data** son otro concepto relacionado con el método de Puntos Función de IFPUG que se incluye en SNAP. En el método de Puntos Función no se miden funcionalmente pero en el método de SNAP sí, ya que forman parte del componente no funcional de los requisitos de usuario. La forma de hacerlo es contar un único grupo lógico adicional para todas las tablas Code Data que tengamos en el sistema.

En un sistema, debido a razones técnicas de validación, un proceso accede a una tabla con los códigos y nombres de los aeropuertos y a otra tabla con los códigos y nombres de ciudades. ¿Cuántos grupos lógicos se deben contar por esta interacción?

1 Grupo Lógico como Code Data que incluye los Aeropuertos y las Ciudades.

La complejidad de este grupo lógico depende de los RET que identifiquemos y éstos coinciden con los distintos subtipos de Code Data que existan en el sistema:

- Sustitución
- Estáticos
- Lista de valores válidos.

Contamos el número y ese será el número de RET del grupo lógico identificado.

Los DET (Data Element Type o Tipo de Elemento de Datos) tienen la misma definición que en el método de Puntos Función de IFPUG salvo que además se miden los que se usen por motivos técnicos y/o code data (código-descripción).

La definición de los FTR (File Type Referenced o Tipos de Fichero Referenciados) es la misma que en Puntos Función salvo que ahora se puede contar un grupo lógico adicional si se accede a alguna tabla Code Data (Código Descripción) pero sólo se puede contar uno, no importa el número de tablas a las que se acceda.

El concepto **Límite** de la aplicación no se ve modificado pero dentro del límite se introduce una subdivisión para intentar medir los movimientos internos de datos, es lo que se conoce como Partición.

La **Partición** es un concepto que intenta reflejar el esfuerzo extra que debe realizarse, en ciertas funciones, para comunicar partes diferenciadas de una aplicación. Una partición es un conjunto de funciones del software en estudio que comparten un criterio y unos valores de medición homogéneos. Las particiones permanecen siempre dentro de los límites de la aplicación y no se superponen unas a otras. Ejemplos de particiones son:

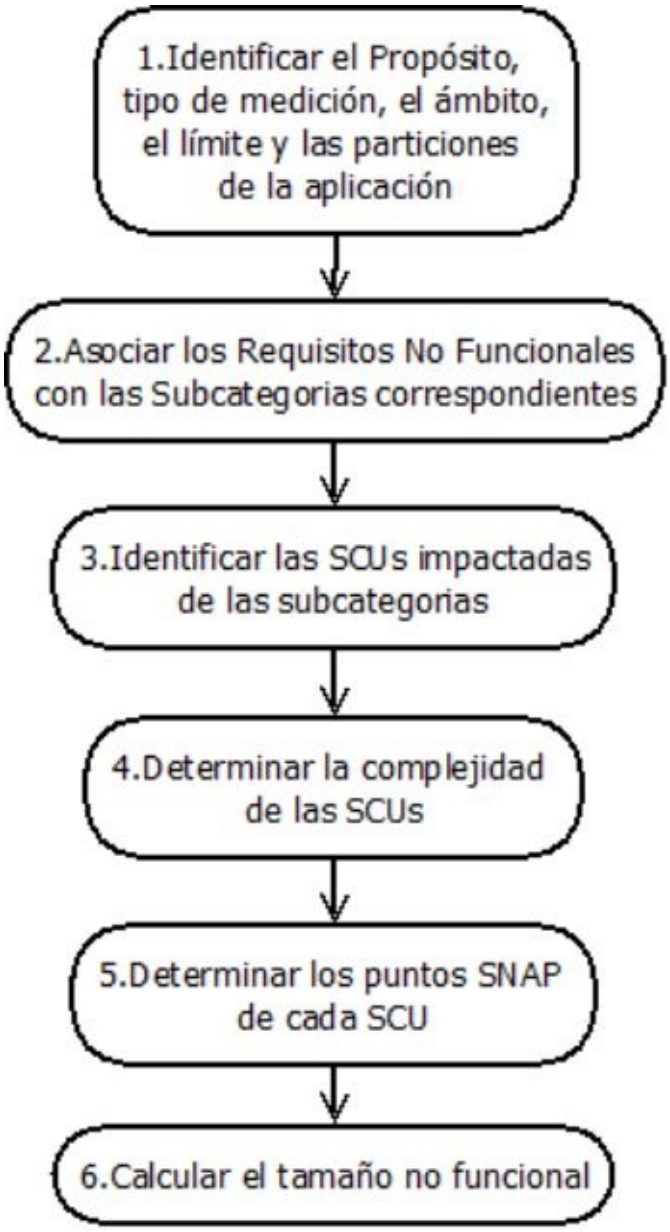
En una aplicación cliente servidor, la parte cliente es una partición y la parte servidor es otra partición.

La aplicación completa puede ser una única partición

PROCESO

No debemos olvidar que el método de SNAP está pensado para ser el complemento del método de IFPUG y, por ello, para que una valoración SNAP sea compatible con una valoración IFPUG debe utilizarse la misma frontera/límite de la aplicación/aplicaciones, es decir no debemos utilizar dos fronteras de aplicación diferentes para el mismo sistema, una para la medición funcional y otra para la medición no funcional.

Los pasos del método SNAP son los siguientes:



1. **Identificar el propósito, el tipo de medición, el ámbito, el límite de la aplicación y las particiones.**

Establecemos cual es el objetivo de nuestra medición. En función del objetivo que tengamos, deberemos utilizar un tipo de medición: *de proyecto, de mejora o de aplicación*. Sabiendo cual es nuestro objetivo y el tipo de la medición podemos establecer el alcance de nuestra medición, qué debemos medir y con él se establecerán los límites de las aplicaciones impactadas. Por último, si aplica, se podrán determinar las distintas particiones de las cuales se compone cada aplicación. Si no se identifica

ninguna, se contaría una única partición.

2. **Asociar los requisitos No Funcionales con sus correspondientes Subcategorías.**

Cada requisito no funcional se mapea con su correspondiente subcategoría.

3. **Identificar las SCU.**

Una vez que tengamos las subcategorías deberemos saber cuáles son las SCU que están impactadas ya que son éstas las que nos darán la medida del impacto No funcional.

4. **Determinar la complejidad de las SCU.**

Por cada una de las Unidades de Conteo de SNAP que hemos identificado en el punto anterior, deberemos analizar y revisar los parámetros por los que se mide su complejidad. Cada subcategoría tiene asociada una SCU y cada SCU tiene asociado unos parámetros a través de los cuales calcular su complejidad.
Una vez determinados los parámetros y contabilizados, podremos calcular la complejidad de la unidad de conteo.

5. **Determinar los puntos SNAP de cada SCU.**

Con la complejidad y las tablas definidas en el método calculamos los puntos SNAP de cada SCU.

6. **Calcular el tamaño no funcional.**

Una vez identificadas todas las SCU y medidas sus complejidades en Puntos SNAP, las sumamos todas para obtener el valor del tamaño no funcional.

CATEGORIAS

Hay 4 categorías que agrupan el conjunto de las 14 subcategorías definidas en el método.

A continuación vamos a ver cada categoría y subcategoría relacionando su SCU

y los parámetros de complejidad de la misma.

Relación de Categorías y Subcategorías de SNAP:

- Operaciones con Datos
    - Validación de la Entrada de Datos
    - Operaciones Lógicas y Matemáticas
    - Formateo de Datos
    - Movimientos Internos de Datos
    - Aportando valor añadido a los usuarios a través de la parametrización
- Diseño del Interfaz
    - Interfaces de Usuario
    - Métodos de Ayuda
    - Múltiples métodos de entrada
    - Múltiples métodos de salida
- Entorno Técnico
    - Múltiples plataformas
    - Tecnología de la Base de Datos
    - Procesos Batch
- Arquitectura
    - Software basado en componentes
    - Múltiples interfaces de entrada/salida

CATEGORIA 1 - OPERACIONES CON DATOS

Esta categoría describe como son procesados los datos dentro de cada SCU para cumplir con los requerimientos no funcionales de la aplicación.

Sus subcategorías son:

1.1 Validación de la Entrada de Datos

Definición: Son requisitos no funcionales que o bien limitan los datos posibles a introducir en una entrada o bien validan los datos introducidos

SCU: Proceso elemental

Complejidad:

- Nivel de anidamiento de las validaciones que se realicen: el número de bloques if-else / while / for que se encadenen unos dentro de otros
- Número de DET necesarios para la validación.

1.2 Operaciones lógicas y matemáticas

Definición: Son requisitos no funcionales encaminados a contabilizar decisiones lógicas complejas, operaciones booleanas y/o operaciones matemáticas complejas. La complejidad viene determinada por el uso de 1 o más algoritmos durante el proceso definido.

SCU: Proceso elemental

Complejidad:

- Número de FTR que accede el proceso elemental
- Número de DET necesarios para el procesamiento

1.3 Formateo de Datos

Definición: Son requisitos no funcionales relativos a la estructura, el formato o la información administrativa dentro de una transacción

SCU: Proceso elemental

Complejidad:

- Complejidad de la transformación en función de lo complejo que sea el procesamiento se cataloga en complejidad: baja, media o alta.

- Número de DET que se procesan.

1.4 Movimientos Internos de datos

Definición: Son requisitos no funcionales relativos a movimientos de datos de una partición a otra, dentro de los límites de la misma aplicación. Dicho movimiento incluye cierto tratamiento de esos datos.

SCU: Proceso elemental, por cada uno de los cruces de partición que realice

Complejidad:

- Número de FTR que se leen o se actualizan
- Número de DET que se envían o se reciben a través de la partición

1.5 Aportando valor añadido a los usuarios a través de la parametrización

Definición: Se aporta valor de negocio único a través de la adición, cambio o eliminación de datos de referencia o datos de codificación (code data) en la base de datos o en el almacenamiento de datos sin realizar cambios en el software ni en la estructura de la base de datos.

SCU: Proceso elemental impactado, por cada fichero lógico

Complejidad:

- Número de Registros configurados
- Número de atributos únicos involucrados

CATEGORIA 2 - DISEÑO DEL INTERFAZ

Esta categoría tiene en cuenta la experiencia del usuario final y por ello mide el diseño de los interfaces que relacionan al usuario con la aplicación.

Sus subcategorías son:

2.1 Interfaces de Usuario

Definición: Elementos del interfaz de usuario añadidos o configurados que no cambian la funcionalidad del sistema pero si modifican las características no funcionales, como son la accesibilidad, facilidad de aprendizaje, atractivo o accesibilidad

SCU: Conjunto de pantallas definidas por un Proceso elemental

Complejidad:

- La suma de las propiedades únicas configuradas
- Número de elementos de interfaz de usuario impactados

2.2 Métodos de Ayuda

Definición: Es información de soporte o información que se provee al usuario explicando como el software desarrolla su funcionalidad.

SCU: La aplicación evaluada

Complejidad:

- Tipo de ayuda
- Número de elementos de ayuda impactados

2.3 Múltiples métodos de entrada

Definición: Es la capacidad de la aplicación para realizar la funcionalidad aceptando múltiples métodos de entrada

SCU: Proceso elemental

Complejidad:

- Número de DET
- Número de métodos de entrada adicionales

2.4 Múltiples métodos de salida

Definición: Es la capacidad de la aplicación para realizar la funcionalidad utilizando múltiples métodos

de salida

SCU: Proceso elemental

Complejidad:

- Número de DET
- Número de métodos de salida adicionales

CATEGORIA 3 - ENTORNO TÉCNICO

Esta categoría está relacionada con aspectos del entorno donde reside la aplicación. Todos los cambios relacionados tanto con tecnología como con los datos internos que no implican un cambio en la funcionalidad desde una perspectiva de los puntos función.

Sus subcategorías son:

3.1 Múltiples plataformas

Definición: Son las operaciones que trabajan en más de una plataforma: Java, Cobol, varios navegadores, etc.

SCU: Proceso elemental

Complejidad:

- Naturaleza de las plataformas
- Número de plataformas a manejar

3.2 Tecnología de la Base de Datos

Definición: Describe las características y las operaciones que se añaden a la base de datos (o a sentencias para leer/escribir de la base de datos) para cumplir con requisitos no funcionales del sistema: creación de tablas code data, creación/modificación de índices, creación/modificación de tablespaces, creación/modificación de vistas, etc.

SCU: Proceso elemental

Complejidad:

- Complejidad del Fichero Lógico (Número de DET y número de RET)
- Número de cambios en la base de datos

3.3 Procesos Batch

Definición: Se contabilizan en esta categoría procesos batch que no se consideran funciones transaccionales: por ejemplo, no tienen DET que crucen la frontera de la aplicación. Si el proceso ya existe cualquier modificación al mismo puede ser contabilizada en esta categoría si no encaja en alguna de las otras

SCU: Proceso Batch identificado por el usuario

Complejidad:

- Número de FTR leídos o actualizados por el proceso
- Número de DET procesados por el proceso batch

CATEGORIA 4 - ARQUITECTURA

Esta categoría está relacionada con el diseño y las técnicas de codificación utilizadas para construir o mejorar la aplicación.

Sus subcategorías son:

4.1 Software Basado en Componentes

Definición: Piezas de software utilizadas dentro de los límites de la aplicación para integrar software que ya existe o para crear componentes dentro del sistema

SCU: Proceso elemental

Complejidad:

- Tipo del componente: componente de terceros o propio

- Número de componentes únicos que intervienen en el proceso elemental

4.2 Múltiples interfaces de Entrada y Salida

Definición: Se encarga de medir la adición de más interfaces de entrada/salida a la aplicación sin modificar la funcionalidad de la misma

SCU: Proceso elemental

Complejidad:

- Número de DET del proceso elemental
- Número de interfaces adicionales de entrada y salida

Apéndice 1 Bibliografía

Son muchas las referencias que he utilizado a lo largo del desarrollo de éste libro. A continuación ofrezco las más destacadas divididas en tres secciones: artículos consultados, libros, guías y otras obras y Blogs y webs de interés.

Artículos

- ***Métodos de Estimación de Tamaño Funcional Software. Aplicación a Enfoques de desarrollo***

Hichem Labdelaoui

- ***Software Measurement***

Università dell'Insubria. Dipartimento di Scienze Chimiche, Fisiche e Matematiche

Sandro Morasca

- ***Modelo de Gestión de Proyectos Software: Estimación del Esfuerzo de Desarrollo***

Universidad de Concepción. Depto. Ing. Informática y Cs. de la Computación. Noviembre 1995

Marcela P. Varas C.

- ***Puntos por Función. Una métrica estándar para establecer el tamaño del software***

Sergio Eduardo Durán Rubio

- ***Métricas, Estimación y Planificación en Proyectos de Software***

Universidad de Guadalajara.

Otoniel Perez Giraldo

- ***Content Management System Effort Estimation Model based on Object Point Analysis***

Naveen Aggarwal, Nupur Prakash, and Sanjeev Sofat

- ***Measuring Object Oriented Software with Predictive Object Points***

Arlene F. Minkiewicz

- ***An Empirical Test of Object-based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment***

  Rajiv D. Banker, Robert J. Kauffman, Rachna Kumar

- ***Estimating with Enhanced Object Points vs. Function Points***

Erik Stensrud

Libros, guías y otras obras

- ***Practical Software Measurement: Measuring for Process Management and Improvement***

William A. Florac, Robert E. Park, Anita D. Carleton

- ***Goal-Driven Software Measurement —A Guidebook***

Robert E. Park, Wolfhart B. Goethert, William A. Florac

- ***Métricas del Software.***

Luis de Salvador Carrasco

Tema 33. Métricas del Modelo de Análisis.

Tema 34. Métricas del Modelo de Diseño.

Tema 35. Métricas del Código Fuente.

Tema 36. Métricas para Pruebas.

- ***Medida del Tamaño Funcional de Aplicaciones Software***

Universidad de Castilla-La Mancha. Escuela Superior de Informática de Ciudad-Real. Mayo - 1999

Faustino Sánchez Rodríguez

- ***Software Size Measurement: A Framework for Counting Source Statements***

Robert E. Park

- ***Herramientas de Gestión de Proyectos***

Serafin Caridad Simón

Tema 2: Modelos de estimación

- ***Tesis: Un Procedimiento de Medición de Tamaño Funcional para Especificaciones de Requisitos***

Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y Computación. Diciembre 2006

Nelly Condori Fernández

- ***Ingeniería del Software. Un Enfoque Práctico. Quinta edición***

Roger S. Pressman

- ***Software Estimation: Demystifying the Black Art***

Steve McConnell

- ***Function Points Analysis. Training Course***

David Longstreet

- ***The COSMIC Functional Size Measurement Method Version 3.0.1. Mayo 2009***

COSMIC

- ***MK II Function Point Analysis Counting Practices Manual Version 1.3.1. September 1998***

UKSMA Metrics Practices Committee

- ***Function Point Counting Practices Manual Release 4.3.1. Enero 2009***

IFPUG

- ***NESMA - FPA for Software Enhancement v2.2.1. 2009***

NESMA

- ***Software Non-functional Assessment (SNAP) Assessment Practices Manual v2.1. Abril 2013***

IFPUG

- ***Functional Size Measurement Method FiSMA 1.1 2008***

FiSMA

- ***Simple Function Point Functional Size Measurement Method Reference Manual SiFP-01.00-RM-EN-01.01 Marzo 2014***

SiFPA

- ***Simple Function Point Functional Size Measurement Method Esempi di applicazione del metodo SiFP-01.00-EX-IT-01.00 Noviembre 2013***

SiFPA

Blogs y webs de interés

- **El Laboratorio de las TI** http://www.laboratorioti.com

Es el blog donde podéis seguir todos los temas de los que escribo sobre todo relativos a estimaciones, mediciones, gestión de proyectos, innovación, etc.

- ***COSMIC*** http://www.cosmicon.com

Web oficial de COSMIC donde puedes encontrar información relativa al método COSMIC-FFP y sobre otros temas de métricas del software.

- ***FiSMA*** http://www.fisma.fi/in-english

Web oficial de la asociación finesa de métricas del software donde puedes encontrar información relativa al método FiSMA.

- ***IFPUG*** http://www.ifpug.org

Web oficial de IFPUG donde puedes encontrar documentación sobre el método FPA de IFPUG, sobre el método SNAP y sobre más temas relacionados con la medición y la estimación.

- ***ISBSG*** http://www.isbsg.org

Base de datos histórica con información de estimaciones y mediciones finales de proyectos reales de múltiples industrias y compañías.

- ***NESMA*** http://www.nesma.nl

Web oficial de la asociación holandesa de métricas del software con información sobre su método del mismo nombre y con documentos realizados por ellos sobre otros temas.

- ***SiFPA*** http://www.sifpa.org/en/index.htm

Web oficial de la asociación de los Puntos Función Simples con información sobre este método.