Universidad de Castilla-La Mancha
Escuela Superior de Informática de Ciudad-Real

Mayo - 1999

Medida del Tamaño Funcional

de Aplicaciones Software

Autor ..... **Faustino Sánchez Rodríguez.**
Asignatura ..... Planificación y Gestión de Sistemas de Información.
Curso ..... 4º Ingeniería Informática.
Profesor ..... Francisco Ruiz González.

Índice de Contenidos

1. INTRODUCCIÓN. 4

2. DEFINICIÓN DE CONCEPTOS Y CLASIFICACIÓN DE LAS MÉTRICAS DEL SOFTWARE. 6

¿QUÉ ES UNA *MÉTRICA*? 6
¿QUÉ ES UNA *MÉTRICA DEL SOFTWARE*? 6
¿QUÉ ES *MEDIR*? 6
¿QUÉ ES *MEDIR EL SOFTWARE*? 6
¿QUÉ ES EL *TAMAÑO FUNCIONAL*? 7
CLASIFICACIÓN DE LAS MÉTRICAS DEL SOFTWARE: 7

3. LOS PUNTOS DE FUNCIONALIDAD (FUNCTION POINTS) 8

3.1. INTRODUCCIÓN: 8
3.2. UN POCO DE HISTORIA... 9
3.3. EL ANÁLISIS DE PUNTOS FUNCIÓN (FPA): 10
3.4. EL MODELO FUNCIONAL Y SU PAPEL EN EL ANÁLISIS DE PUNTOS FUNCIÓN: 18
3.4.1. Componentes del Modelo Funcional: *18*
3.4.2. Representación del modelo funcional: *21*
3.4.3. Extensión del modelo funcional para su medida en Puntos Función: *21*
3.5. APLICACIÓN DEL TAMAÑO FUNCIONAL EN LA ESTIMACIÓN DE PROYECTOS SOFTWARE: 25
3.5.1. Estimación del esfuerzo de desarrollo: *25*
3.5.2. Utilización de los Puntos Función en la estimación del esfuerzo: *26*
3.5.3. Estimación de la duración del proyecto: *29*
3.6. AUDITORÍA DEL CONTEO DE PUNTOS FUNCIÓN: 29
3.6.1 Acumulación y Evaluación de Evidencias: *30*
3.6.2. Tipos de Informes: *31*
3.6.3. Preguntas clave en la auditoría del conteo de Puntos Función: *32*
3.7. LOS PUNTOS FUNCIÓN NO SON LA MÉTRICA PERFECTA: 33

4. LOS PUNTOS CARACTERÍSTICA (FEATURE POINTS). 34

4.1. INTRODUCCIÓN: 34
4.2. PROCESO DE CONTEO DE PUNTOS CARACTERÍSTICA: 34
4.3. LOS ALGORITMOS EN EL PROCESO DE CONTEO DE PUNTOS CARACTERÍSTICA. 35
4.4. PUNTOS FUNCIÓN VS PUNTOS CARACTERÍSTICA. 36

5. LA MÉTRICA *BANG*. 37

5.1. INTRODUCCIÓN: 37
5.2. COMPONENTES ELEMENTALES DEL MODELO DE ESPECIFICACIÓN: 37
5.3. CLASIFICACIÓN DE LOS SISTEMAS: 39
5.4. FORMULACIÓN DE LA MÉTRICA BANG: 40
A. Para sistemas orientados a la función: *40*
B. Para sistemas orientados a los datos: *42*
C. Para sistemas híbridos: *42*

6. EL MODELO DE ESTIMACIÓN COCOMO II. 43

6.1. INTRODUCCIÓN: 43
6.2. EL MODELO COCOMO 81: 43
6.3. EL MODELO COCOMO II: 44
6.4. LOS PUNTOS OBJETO COMO MEDIDA DEL TAMAÑO FUNCIONAL: 44

7. PUNTOS FUNCIÓN COMPLETOS (FULL FUNCTION POINTS) 47

7.1. INTRODUCCIÓN: 47
7.2. EL SOFTWARE DE TIEMPO REAL: 47
7.3. LIMITACIONES DEL ANÁLISIS FPA CON SISTEMAS DE TIEMPO REAL: 47

7.4. EXTENSIÓN DEL MODELO FPA PARA SISTEMAS DE TIEMPO REAL: 48
7.4.1. Tipos de funciones en el análisis FFP: 48
7.4.2. Asignación de Puntos Función Completos a los componentes del sistema: 49
7.4.3. Estimación del esfuerzo de desarrollo: 50

8. LOS PUNTOS FUNCIÓN 3D. 51

9. EL ESTÁNDAR ISO/IEC 14143 PARA MEDIDA DEL TAMAÑO FUNCIONAL. 52

9.1. INTRODUCCIÓN: 52
9.2. EL ESTANDAR ISO/IEC 14143: 52
Parte 1: Definición de conceptos para Medida del Tamaño Funcional (FSM) 53
Parte 2: Adaptación de métodos de medida del software al estándar ISO/IEC 14143-1. 55
Parte 3: Verificación de métodos FSM. 56
Parte 4: Modelo de referencia para FSM. 56
Parte 5: Definición de Dominios Funcionales. 56

ANEXO I: REFERENCIAS WEB. 57

ANEXO II: REFERENCIAS BIBLIOGRÁFICAS. 58

ANEXO III: GLOSARIO DE TÉRMINOS Y ACRÓNIMOS. 60

Índice de Figuras

Figura 1. Clasificación de las métricas del software. 7
Figura 2. Tamaño, en Puntos Función, de algunos tipos de aplicaciones software. 9
Figura 3. Representación gráfica de un modelo funcional. 22
Figura 4. Descripción textual de un modelo funcional. 23
Figura 5. Catalogación de los sistemas software, en función de los ratios DEO / FP y RE / FP. 40
Figura 6. Funciones de Control Transaccional relacionadas con un sistema de control. 49

Índice de Tablas

Tabla 1. Grados de relevancia de las GSC's en el sistema. 12
Tabla 2. Las 14 características generales del sistema, para el cálculo de Puntos Función. 13
Tabla 3. Clasificación de los componentes del sistema, según el modelo COCOMO II. 15
Tabla 4. Conteo de Puntos Función. 17
Tabla 5. Atributos asignados a transacciones, y posibles valores. 20
Tabla 6. Características relevantes del conjunto de ficheros lógicos de un sistema. 24
Tabla 7. Tamaño funcional de una aplicación, desglosado según tipo de componente y complejidad del mismo. 24
Tabla 8. Pros y contras de las técnicas de estimación del esfuerzo. 25
Tabla 9. Utilización del FPA en la estimación del esfuerzo de desarrollo. 26
Tabla 10. Atributos para comparación de proyectos. 28
Tabla 11. Conteo de Puntos Característica. 35
Tabla 12. Ratios comparativos entre Puntos Función y Puntos Característica, para distintos sistemas. 36
Tabla 13. Componentes elementales que han de contabilizarse para aplicar la métrica Bang. 38
Tabla 14. Categorías y factores de corrección, según la complejidad de las primitivas funcionales. 41
Tabla 15. Factores de corrección (pesos) para los objetos de un sistema, ...42
Tabla 16. Clasificación de los objetos del sistema (pantallas de interfaz e informes). 45
Tabla 17. Pesos (en Puntos Objeto), en función de la complejidad de los objetos del sistema. 46
Tabla 18. Ratios de productividad según equipo y entorno de desarrollo. 46
Tabla 19. Tipos de funciones en la técnica FFP. 49
Tabla 20. Números de FFP's asignados a las funciones de control transaccional, ...50

1. Introducción.

Hoy en día es impensable, en cualquier negocio, mejorar de manera consistente sin disponer de una base cuantitativa. La **medición** es una parte esencial del proceso de mejora de cualquier actividad humana. Existe un ciclo de mejora que consta en *Medir - Analizar - Tomar Acciones* y que debe ser aplicado a procesos, productos y clientes, con el objeto de determinar conceptos como innovación, productividad, costes, calidad del producto, servicio al cliente...

Con estos objetivos se pueden definir ***Métricas del Software*** para obtener una visibilidad precisa de los procesos de desarrollo y mantenimiento del software, así como de los productos que conforman el Sistema de Información de una Organización. De esta forma el *management* de los Sistemas de Información dispondrá de una vía cuantitativa, tal como lo tienen desde hace ya muchos años los directores de industrias tradicionales, permitiendo responder a preguntas tales como:

- ¿Cuánto cuesta el desarrollo de un nuevo sistema?
- ¿Cuándo acabará?
- ¿Cuál es la innovación de tal producto?
- ¿Qué grado de mantenibilidad tiene esta aplicación? ¿Tengo que subir o bajar el coste de los subcontratistas?
- El producto que vamos a lanzar al mercado, ¿tiene al menos una fiabilidad superior al 95%? ¿Qué fiabilidad tiene, concretamente?
- Hay que desarrollar un nuevo producto que es bastante parecido a uno ya existente. ¿Tiene este último una reusabilidad y portabilidad suficientes para que valga la pena tomarlo como base, o es mejor hacerlo de nuevo?. En todo caso, ¿qué componentes o partes de la aplicación antigua son lo suficientemente portables y reusables para adaptarlos al nuevo sistema?
- ¿Qué grado de aceptación tienen nuestros productos para los clientes? ¿Qué nivel de calidad perciben? ¿En qué grado satisface el producto las necesidades esperadas por los clientes?

Es importante señalar que todas estas preguntas, y cualquier otra típica de los directores de las organizaciones, tienen una respuesta cuantitativa. De todas ellas se deduce que ***medir el software*** **no sólo es útil, sino necesario**. Y es necesario para poder conocer en todo momento el estado o la situación en la que se encuentran los proyectos, los productos, los procesos y los recursos. En otras palabras, debemos controlar los proyectos, productos, procesos y recursos, y no sólo ejecutarlos, desarrollarlos, gestionarlos o administrarlos (respect.).

Pero cada intención de medir un aspecto particular del software tiene que estar fundamentada en una razón concreta, precisa y bien definida. No basta decir que hay que medir para ganar control sobre un proyecto, o para mejorar un proceso determinado. Los objetivos que se pretendan con la medición deberán ser específicos y acordes con las necesidades y requerimientos tanto de los usuarios como de los desarrolladores y directivos.

En definitiva, la gestión precisa de cualquier proyecto, proceso o producto requiere de la cuantificación y la medida, y para ello, las métricas del software proporcionan la base cuantitativa que sustituye a los procedimientos informales y *ad-hoc* que se utilizaban anteriormente.

Los resultados derivados de la aplicación de estas métricas podrán ser utilizados como datos de entrada en modelos de estimación de costes, productividad, etc... En este ámbito de aplicación, una de las medidas principales es la del **tamaño** de un producto software. Básicamente, existen dos medidas para cuantificar el tamaño:

- Las **medidas técnicas**, utilizadas en la medición de productos y procesos software, desde el punto de vista del desarrollador de esos productos y/o procesos.
- Las **medidas funcionales**, utilizadas en la medición de productos y procesos software, desde el punto de vista del usuario, siendo independientes de cualquier aspecto técnico y de cualquier decisión de implementación. Pueden ser usadas, por tanto, para comparar la productividad de diferentes técnicas y tecnologías.

El presente trabajo tratará en profundidad este último tipo de métricas, las **funcionales,** que dan lugar a un conjunto de técnicas utilizadas en la medición y cuantificación del **tamaño funcional** de las aplicaciones software (en inglés, *Functional Size Measurement* (FSM)).

Tras una breve **definición de conceptos** (*Apartado 2*), se pasa a estudiar en detalle la técnica de **Análisis de Puntos Función** (FPA) en el *Apartado 3*. El método FPA es, quizás, el más conocido y utilizado para medir el tamaño funcional de sistemas software; pero, a lo largo del tiempo, han ido surgiendo nuevas variantes con la intención de adaptar este método a las características y necesidades de los productos y procesos actuales (nuevas metodologías, nuevas tecnologías, etc...). Los *Apartados 4, 7* y *8* estudian algunas de estas variantes: los **Puntos Característica**, los **Puntos Función Completos** y los **Puntos Función 3D.** Dejando a un lado las técnicas basadas en los Puntos Función de Albrecht, se estudia en el *Apartado 5* la aproximación de Tom DeMarco a la medida del tamaño funcional: la **métrica Bang**. El *Apartado 6* introduce al lector en la revisión y adaptación del más conocido de los modelos de estimación de costes: el modelo **COCOMO II** que, basándose en un proceso de cálculo muy similar al del análisis FPA, calcula la funcionalidad de un sistema en términos de **Puntos Objeto.** Finalmente, el *Apartado 9* hace referencia al único estándar internacional que contempla y normaliza la medida del tamaño funcional de aplicaciones software: el estándar ISO/IEC 14143.

Se han incluido también varios anexos para dar cabida a un **Glosario de Términos y Acrónimos**, **Referencias Bibliográficas** y **Referencias a páginas Web,** que proporciona detallada información sobre el tema que nos ocupa: las *métricas para Medida del Tamaño Funcional*.

2. Definición de conceptos y clasificación de las métricas del software.

¿Qué es una *métrica*?

Para dar respuesta a esta pregunta, haremos uso de una cita enunciada por Paul F. Lazarsfeld, profesor de la prestigiosa Universidad de Columbia. En sus clases, acostumbraba a decir a sus alumnos:

"En el momento en que asocias un número con una idea, aprendes algo nuevo.
Entonces, sabes algo más de lo que sabías antes".

Entonces, una *métrica* es el número que se asocia a una idea. Concretando... una métrica es la variable o medida de ciertos aspectos cuantitativos de un sistema. Por ejemplo, para un sistema software, dichos aspectos cuantitativos serían el alcance, el tamaño, el coste, los riesgos, etc...

Pero tan importante como saber lo que es una métrica, es saber lo que no es. Una métrica no es meramente una observación expresada en términos numéricos. Por ejemplo, supongamos la siguiente afirmación: *"El 95% de los programadores coinciden en que ya se ha desarrollado el 50% de la aplicación"*. Esta observación no constituye una métrica. Para que lo fuera debería implicar un cierto proceso de *medición* de algún aspecto cuantitativo, como se comentó anteriormente.

¿Qué es una *métrica del software*?

La definición más común de *métrica del software* es la siguiente: *"Una métrica software es un atributo del entorno de desarrollo del software, derivada de la medida de los atributos de ciertos componentes del software"*. Un atributo es una cualidad, una propiedad o una característica de un objeto. En el entorno de desarrollo del software, el tamaño, el coste y el esfuerzo son algunos de los atributos del proyecto software.

Una definición más precisa de *métrica del software* es la aportada por el estándar IEEE 610.12 sobre glosario de términos utilizados en Ingeniería del Software: *"Una métrica del software es una medida cuantitativa del grado en el que un sistema, componente o proceso dispone de un atributo dado"*. Desde otro punto de vista bien distinto, Fenton aporta la siguiente definición [FENTO97]: *"Una métrica software es una correspondencia entre uno o más atributos del entorno de desarrollo del software, y cualquier otro atributo"*.

¿Qué es *medir*?

Según la última definición de *métrica del software* que hemos visto, *medir* es el proceso a través del cual se asigna o hace corresponder un número a los atributos de los objetos. Dichos números son asignados siguiendo una serie de reglas claras y bien definidas. Es necesario también disponer de una descripción precisa de los atributos de los objetos que pretendermos *captar* numéricamente.

¿Qué es *medir el software*?

Medir el software es el proceso a través del cual podemos *cuantificar* el software, medir el conjunto de recursos involucrados en el ciclo de desarrollo y medir incluso el propio proceso de desarrollo. Esto incluye, por tanto, elementos que son directamente *medibles* (como las líneas de código), y otros que no lo son directamente, sino que son calculados a través de fórmulas o ecuaciones (como el esfuerzo de desarrollo o el coste del proyecto).

¿Qué es el *tamaño funcional*?

La norma ISO/IEC 14143 (ver *Apartado 9*) define *tamaño funcional* como *"el tamaño de una aplicación software derivado de la cuantificación de los Requisitos Funcionales del Usuario"*. Y define *Requisitos Funcionales de Usuario* como *"un subconjunto de los requisitos de usuario que representan las prácticas y los procedimientos de usuario que el software debe realizar para completar sus necesidades, y excluyéndose cualquier requisisto técnico y de calidad"*.

En otras palabras, el tamaño funcional de una aplicación software es la medida de la funcionalidad que implementa una aplicación, desde el punto de vista de los usuarios, e independiente de cualquier aspecto técnico y de cualquier decisión de implementación.

Clasificación de las métricas del software:

Existen muchas formas de clasificar las métricas del software, distintas unas de otras según los autores. Las métricas del software pueden ser:

- **Directas** o **Indirectas.**
- **Internas** o **Externas.**
- **Simples** o **Complejas.**
- **Primitivas** o **Calculadas.**
- **Primarias** o **Secundarias.**
- **Públicas** o **Privadas.**
- **De Proceso, de Producto** o **de Proyecto.**
- etc...

Pero de entre todas estas posibles clasificaciones, comentaremos sólo las *métricas directas* y las *indirectas* del producto. Entre las métricas directas del producto se encuentran las líneas de código producidas (LDC), la velocidad de ejecución, etc... Y entre las métricas indirectas del producto se encuentran la funcionalidad, calidad, complejidad, eficiencia, etc...

El coste y el esfuerzo requeridos para la construcción de software, el número de líneas de código y otras medidas directas pueden ser relativamente fáciles de obtener. Sin embargo, la calidad, la funcionalidad, la complejidad del software, son más difíciles de evaluar y sólo pueden medirse indirectamente.

Si con alguna clasificación de las métricas hemos de quedarnos, lo haremos con la propuesta por Roger S. Pressman [PRES97]. Pressman clasifica el campo de las métricas tal y como se muestra en la figura adjunta. Como vemos, distingue 6 categorías o grupos de métricas distintos:

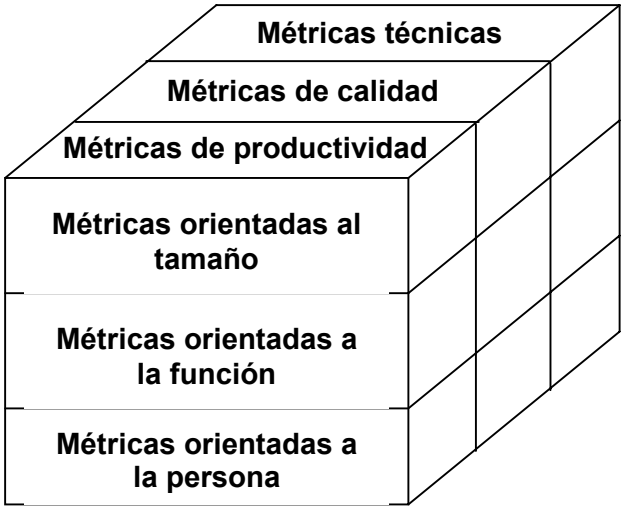

Figura 1. Clasificación de las métricas del software.

- **Métricas técnicas**, centradas en las características del software más que en su proceso de desarrollo.
- **Métricas de calidad**, tanto del software desarrollado como de la efectividad del proceso de ingeniería aplicado.
- **Métricas de productividad**, referidas al rendimiento del proceso de desarrollo como función del esfuerzo aplicado.
- **Métricas orientadas al tamaño**, que miden de forma directa el software y el proceso por el cual se desarrolla.
- **Métricas orientadas a la función**, que se centran en la *funcionalidad* o *utilidad* del programa, y que estudiaremos en detalle en los siguientes apartados.
- **Métricas orientadas a la persona**, que aportan información sobre la forma en que la gente desarrolla software.

La clasificación que hemos visto nos da una idea bastante clara del contexto en el que están ubicadas las métricas orientadas a la función, objeto de estudio de este trabajo.

3. Los Puntos de Funcionalidad (Function Points).

3.1. Introducción:

Hoy en día resulta muy complejo contestar a preguntas como:

- ¿Qué funcionalidad proporciona una aplicación software concreta?
- ¿Puede medirse dicha funcionalidad y compararla con la de otro producto similar?
- ¿Cuál es la productividad del equipo de desarrollo? ¿Y la del de mantenimiento?
- ¿Qué precisión tienen las estimaciones que se realizan en los proyectos de desarrollo y mantenimiento del software?
- ¿Está mejorando la calidad del software que se realiza? Si es así, ¿cómo se sabe?
- En nuestra empresa de desarrollo, ¿tenemos personal suficiente para desarrollar y mantener el software especificado? ¿Quizás sobra gente? ¿Cómo puede demostrarse?
- ¿Cuál es el valor actual de nuestras inversiones en software? ¿Cuanto costaría reponer nuestra cartera de aplicaciones?
- Al subcontratar tareas de desarrollo o mantenimiento software, ¿se sabe de antemano cual será el coste real del trabajo?

Para poder responder a estas y a otras preguntas es necesario disponer de un proceso de medida, tanto de los procesos, como de los productos software que se desarrollan y de las aplicaciones que se mantienen.

Los Puntos Función proporcionan una medida objetiva, cuantitativa y auditable del tamaño de las aplicaciones, desde el punto de vista de los requisitos especificados por el usuario final de la aplicación. También son un medio de entendimiento entre lo que el usuario quiere y lo que al final se le suministra. En consecuencia, su valoración se deriva a partir de los requisitos funcionales que la aplicación debe satisfacer, modelos de datos, definición de pantallas e interfaces gráficos y diagramas de análisis.

Los Puntos Función constituyen una técnica de medida del software, simple de obtener pero muy potente en sus resultados. Esta potencia radica en que del valor de la medida en Puntos Función se derivan un conjunto de métricas esenciales para la gestión de la productividad, la calidad y el coste del software. Con estas medidas, registradas en distintas fases del ciclo de vida, se puede llevar a cabo un análisis exhaustivo de su evolución y, por tanto, del control de la productividad, la calidad y los costes asociados, a lo largo del tiempo. De esta forma, y almacenando en un registro histórico de datos el valor en Puntos Función de cada uno de los proyectos realizados, podremos disponer de una sólida base para futuras estimaciones del coste y duración de los proyectos, información altamente valiosa para la dirección de las organizaciones.

Actualmente, este método de estimación es uno de los más usados (sobre todo en EE.UU. y Europa). De hecho, se calcula que el número total de proyectos software medidos en Puntos Función sobrepasa ya los 100.000 en todo el mundo. Sin embargo, esta cifra supone menos del 1% de las aplicaciones software en uso hoy en día. Lo cierto es que la mayoría de las organizaciones desarrollan software, pero sin realizar ningún tipo de medición útil sobre el mismo. Algunos estudios revelan que la mayoría de las pequeñas empresas de desarrollo de software (entendiendo como tales aquéllas con menos de 100 profesionales en la materia) no utilizan ni la estimación en Puntos Función, ni cualquier otro método de estimación. Sólo las grandes organizaciones (aquéllas con más de 10.000 personas involucradas en los procesos de desarrollo, testeo y mantenimiento de aplicaciones software) son las que están haciendo un mayor uso de los Puntos Función, como método preciso de estimación y medida.

Es también interesante fijarse en el tamaño de algunas aplicaciones software de uso habitual. Por ejemplo, el procesador de textos que se ha utilizado para escribir este documento puede tener perfectamente más de 400.000 líneas de código y unos 3500 Puntos Función. Un administrativo, en el trabajo diario con aplicaciones ofimáticas, puede utilizar quizás más de 25.000 Puntos Función en forma de hojas de cálculo, bases de datos, procesadores de texto, herramientas estadísticas, etc... Todas estas aplicaciones corren bajo el control del sistema operativo, el cual puede llegar incluso hasta los 100.000 Puntos Función en tamaño. El siguiente gráfico muestra el tamaño en Puntos Función de algunos tipos de software concretos:

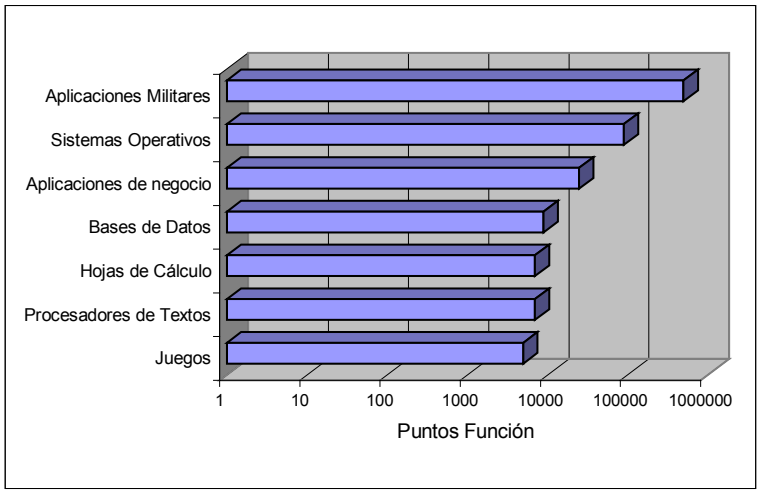

Figura 2. Tamaño, en Puntos Función, de algunos tipos de aplicaciones software.

3.2. Un poco de historia...

A finales de los años '70, Allan J. Albrecht [ALBR79] [ALBR83] desarrolló una unidad de medida capaz de determinar la funcionalidad de los sistemas software. A esta métrica la llamó *"Puntos de Funcionalidad"* o, más sencillamente, *"Puntos Función"*. La teoría de Albrecht se basaba en estudiar, por separado, cinco componentes o características principales del sistema software: las **entradas**, las **salidas**, las **consultas** o peticiones interactivas (cuando el usuario hace una petición al sistema y éste devuelve una respuesta), los **ficheros lógicos internos** (ficheros maestros) y los **ficheros lógicos externos** (interfaces con otras aplicaciones).

Pero el conteo de Puntos Función no se limitaba sólo a contar el número de componentes del sistema, sino que era preciso, además, aplicar ciertos valores representativos de la complejidad de cada elemento. Así, tras muchos ensayos, Albrecht obtuvo empíricamente los factores de peso que debían aplicarse a cada uno de los cinco componentes. Estos factores variaban en función de la complejidad de cada componente, siendo esto indicativo de la mayor o menor dificultad que conllevaba su implementación.

En Octubre de 1979, en Monterey (California), Albrecht presentó por vez primera los resultados obtenidos en sus estudios [ALBR79], proponiendo el concepto de *Puntos de Funcionalidad* como una nueva métrica de las aplicaciones software.

En 1986 nació la IFPUG (Agrupación Internacional de Usuarios de Puntos Función), cuyos objetivos eran, entre otros, dar soporte a esta nueva técnica y promocionar su uso. En 1987, el gobierno británico adoptó la técnica propuesta por Albrecht como el estándar a utilizar para medir la productividad de las aplicaciones software que se desarrollaran. Más tarde, en 1990, la IFPUG publicó la versión 3.0 del compendio de reglas y criterios para el conteo de Puntos Función: el CPM (*Counting Practices Manual*). Desde Abril de 1995 está en vigor la versión 4.0 de dicho manual.

Actualmente, la Organización Internacional para la Estandarización (ISO), junto con las más importantes asociaciones de usuarios de Puntos Función, están trabajando en la elaboración de lo que será la norma ISO-14143 sobre Medida del Tamaño Funcional de aplicaciones software. Esta norma consta de cinco partes bien diferenciadas y puede consultarse en el *Apartado 9* de este trabajo.

3.3. El Análisis de Puntos Función (FPA):

El Análisis de Puntos Función nos proporcionará una medida objetiva de la funcionalidad de una aplicación software, ayudándonos en la evaluación, planificación, gestión y control de los procesos de desarrollo software. Inicialmente propuesto por Allan Albrecht en 1979 [ALBR79], está basado en la teoría de que las funciones de una aplicación software son la mejor medida de su tamaño. Se trata de un método utilizado en la medición del tamaño del software, desde el punto de vista de los requisitos funcionales que debe implementar. Dichos requisitos funcionales son determinados a partir de las necesidades concretas del usuario/cliente.

Así pues, este método tratará de determinar el número total de Puntos Función de un sistema, basándose únicamente en los documentos técnicos que recogen la especificación de requisitos funcionales. Por lo tanto, una de las características principales de este método es que su aplicación es factible ya desde las primeras etapas del ciclo de vida de la aplicación software, es decir, desde el momento en que se elaboran dichos documentos técnicos.

En definitiva, el Análisis de Puntos Función tratará de determinar, tan pronto como sea posible, la funcionalidad que proporciona a los usuarios una determinada aplicación software. Pero esto no es tarea sencilla; para el conteo de Puntos Función se han de seguir una serie de pasos concretos basados en reglas y criterios específicos definidos por la IFPUG, hasta llegar al resultado final. Dicho resultado no es único, sino que de la aplicación del FPA pueden derivarse otros. A saber...

- Cuenta del número total de Puntos Función sin ajustar.
- Cuenta de Puntos Función para cada componente del sistema (Entradas Externas, Salidas Externas, Consultas Externas, Ficheros Lógicos Internos y Ficheros Externos de Interfaz).
- Factor de Ajuste (VAF) o Factor de Complejidad Técnica.
- Cuenta del número total de Puntos Función ajustados.
- Informes de validación de resultados.

Los datos de partida necesarios para comenzar el proceso de FPA suelen estar localizados en documentos técnicos que recogen, entre otros, la siguiente información:

- Objetivos que se pretenden cubrir, necesidades y requerimientos del usuario.
- Toda la información disponible acerca del sistema actual (si existe).
- Cualquier objetivo específico que se pretenda para el nuevo sistema y restricciones concretas que puedan existir.
- El interfaz gráfico de usuario que se desea.
- Los interfaces existentes con otros sistemas.
- Los modelos de datos físicos y/o lógicos preliminares.

Como comentamos antes, el proceso de FPA consta de varios pasos. En total son ocho:

1. Planificación del conteo de Puntos Función.
2. Recogida de información.
3. Cálculo del Factor de Ajuste (VAF).
4. Inventariado de transacciones lógicas y ficheros lógicos.
5. Clasificación de componentes.
6. Revisión de las 14 características generales del sistema (GSC's).
7. Tabulación de resultados.
8. Validación de resultados.

Pero en el proceso de FPA, antes de comenzar con el primer paso, es necesario realizar una tarea de vital importancia. Se trata de determinar el **contexto de la aplicación** que se va a medir, es decir, determinar dónde está el sistema, cuales son sus fronteras con otros posibles sistemas, y en qué contexto se encuentran todos ellos. En este paso preliminar, tanto el experto en el conteo de Puntos Función como el experto en determinar la funcionalidad del sistema, deberán trabajar juntos para obtener, como resultado, el diagrama de contexto general sobre el que basarse para comenzar el conteo.

En los apartados que siguen a continuación comentaremos en detalle cada uno de los ocho pasos citados anteriormente:

Paso 1: Planificación del conteo de Puntos Función.

El tamaño en Puntos Función de una aplicación puede ser determinado justo después de concluir el proceso de especificación de requisitos, pero también antes de haberlo terminado completamente. En este último caso, debería realizarse un nuevo conteo una vez hayan quedado perfectamente definidos todos los requisitos funcionales. Lógicamente, estimaciones tan tempranas están sujetas a posibles cambios y fluctuaciones, influidas mayoritariamente por posibles cambios en la especificación de requisitos. En definitiva, si el alcance y la envergadura del proyecto cambia en algún momento, el esfuerzo requerido para completarlo también cambiará.

El proceso de FPA también puede aplicarse a cualquier otra etapa del ciclo de vida (diseño, implementación, pruebas, mantenimiento...). Por ello es recomendable que, una vez concluido el proceso de conteo, el resultado sea comparado con otros obtenidos en etapas anteriores. Así podremos tener en cuenta el impacto provocado por la introducción de nuevos componentes en el sistema o por cambios experimentados en cualquiera de ellos. Además, también podremos actualizar apropiadamente el número total de Puntos Función.

Paso 2: Recogida de información.

En el caso de que apliquemos el proceso de FPA antes de finalizar la especificación total de requisitos software, disponer de la siguiente documentación nos resultará de bastante utilidad:

- Objetivos que se pretenden cubrir, necesidades y requerimientos del usuario.
- Toda la información disponible acerca del sistema actual (si existe).
- Cualquier objetivo específico que se pretenda para el nuevo sistema y restricciones concretas que puedan existir.
- Diagrama de contexto del sistema en su conjunto.

Tras las etapas de análisis y diseño del sistema, puede realizarse un nuevo conteo de Puntos Función más preciso. Para ello, sería conveniente disponer de la siguiente documentación:

- Interfaz gráfico de usuario que se desea (formatos de pantalla, cuadros de diálogo y menús de opciones).
- Interfaces existentes con otros sistemas.
- Posibles formularios de entrada de datos al sistema.
- Modelos de datos físicos y/o lógicos preliminares.
- Formato y tamaño de los ficheros del sistema.

Comparando entre sí los resultados en Puntos Función obtenidos antes y después de estas etapas, nos haremos una idea de cómo ha crecido la aplicación en cuanto a funcionalidad, desde la etapa de especificación de requisitos.

Paso 3: Cálculo del Factor de Ajuste (VAF).

El *Factor de Ajuste VAF* (Value Adjustment Factor), también llamado *Factor de Complejidad Técnica*, debería calcularse ya en las primeras ocasiones en las que se realiza el conteo de Puntos Función para, más adelante, ir recalculándolo y actualizándolo (si es necesario), según vayamos obteniendo más información sobre el sistema.

El VAF está basado en 14 características generales del sistema (General System Characteristics ó GSC's) que evalúan la funcionalidad general de la aplicación que se está midiendo. Cada GSC tiene asociada una serie de cuestiones o preguntas acerca de la misma, cuya respuesta ayuda a determinar su grado de importancia dentro del sistema en función de una escala que va de cero (sin influencia) a cinco (esencial), según se muestra en la siguiente tabla:

Valor:	**0**	**1**	**2**	**3**	**4**	**5**
Significado:	Sin influencia	Incidental	Moderado	Medio	Significativo	Esencial

Tabla 1. Grados de relevancia de las GSC's en el sistema.

Al evaluar cada GSC puede entreverse una cierta subjetividad. Por ejemplo: ¿cuándo una GSC es *esencial* para el sistema?, ¿Cuándo puede decirse que su importancia es sólo *incidental*? A este respecto, la IFPUG aporta una serie de criterios de evaluación detallados, al objeto de eliminar la máxima subjetividad posible [IFPUG95].

En la tabla 2 se muestran las 14 características generales comentadas anteriormente, junto con las preguntas típicas que han de formularse acerca de ellas.

Una vez que se ha determinado la influencia de cada GSC en el sistema (valor entre 0 y 5), se utiliza la siguiente fórmula para obtener el valor del VAF:

$$VAF = 0.65 + 0.01\sum_{i=1}^{14} F_i$$

...siendo F_i el valor adjudicado a cada GSC.

Como vemos, el VAF puede variar entre 0.65 (si cada F_i vale 0, es decir, si las GSC's no tienen ninguna influencia en el sistema) y entre 1.35 (si cada F_i vale 5, es decir si todas las GSC's son esenciales para el sistema). Más adelante veremos cómo el VAF calculado se utiliza como factor corrector del conteo total de Puntos Función.

Paso 4: Inventariado de transacciones lógicas y ficheros lógicos.

Este paso consiste en determinar cuáles son los componentes del sistema a medir, de interés para el conteo de Puntos Función. El FPA descompone los sistemas en componentes más pequeños, de tal manera que los usuarios, desarrolladores y administradores puedan entenderlos y analizarlos mejor. En el ánbito de los Puntos Función, los sistemas están divididos en cinco componentes básicos:

- **Entradas Externas:** Cada Entrada Externa es un proceso elemental a través del cual se permite la entrada de datos al sistema. Estos datos provienen bien de una aplicación ajena al sistema, o bien del usuario, el cual los introduce a través de una pantalla de entrada de datos (órdenes concretas, nombres de ficheros, selecciones de menús, etc...). No se incluyen las consultas o

Características Generales del Sistema (GSC's)		Cuestiones	Ejemplo
1	Comunicación de datos	¿Qué necesidades de comunicación requiere el sistema para transferencia o intercambio de información?	Una aplicación para el sector bancario, donde se requieren numerosas transacciones monetarias.
2	Procesamiento de datos distribuido	¿Existen funciones de procesamiento distribuido? ¿Cómo son manejados los datos distribuidos?	Un motor de búsqueda en Internet, donde el procesamiento está distribuido en decenas de máquinas.
3	Rendimiento	¿Es importante el tiempo de respuesta? ¿Es crítico el rendimiento?	Una aplicación para el control del tráfico aéreo, que debe proporcionar continuamente información precisa sobre la posición y rumbo de los aviones.
4	Uso del hardware existente	¿En qué medida se está utilizando la plataforma hardware en donde se ejecutará la aplicación?	Un sistema para matrículas en una universidad, donde concurren cientos de alumnos al mismo tiempo.
5	Transacciones	¿Con qué frecuencia se ejecutan las transacciones? (diariamente, semanalmente, mensualmente, etc...)	Una aplicación para el sector bancario, donde deben realizarse millones de transacciones durante la noche.
6	Entrada de datos interactiva	¿Requiere el sistema entrada de datos interactiva? ¿Cuánta información se captura on-line? (en %)	Un programa en el que los datos de entrada provienen de papeles o formularios impresos.
7	Eficiencia	¿Se diseñó la aplicación pensando en que fuera eficiente y fácilmente utilizable por el usuario?	Un programa de análisis financiero utilizado por el directivo de una empresa, capaz de orientarle y asesorarle.
8	Actualizaciones on-line	¿Cuántos ficheros Ficheros Lógicos Internos se actualizan interactivamente (por medio de transacciones on-line)?	Una aplicación para reserva de billetes, en la que deben bloquearse y modificarse ciertos registros en las BB.DD.'s para evitar que un mismo asiento sea vendido dos veces.
9	Complejidad de procesamiento	¿Existe mucha carga en cuanto a procesami. lógico y/o matemático? ¿Es complejo el procesamiento interno?	Un sistema para diagnóstico médico, el cual realiza costo-sas operaciones de decisión lógica hasta obtener un result.
10	Reusabilidad	¿Se desarrolló la aplicación para cumplir las necesidades de más de un usuario? ¿Se ha diseñado el código para ser reutilizable?	Un procesador de textos en el que, por ejemplo, su barra de menús puede utilizarse desde una hoja de cálculo, un generador de informes de una base de datos, etc...
11	Facilidad de conversión e instalación	¿Cómo son de difíciles la conversión y la instalación? ¿Se ha incluido en el diseño la conversión y la instalación?	Cualquier aplicación de propósito general, de tal forma que cualquier persona pueda realizar la instalación fácilmente.
12	Facilidad de operación	¿Requiere el sistema copias de seguridad y de recuperación fiables? ¿Cómo son de efectivos y qué grado de automatización tienen los procesos de arranque, copia de seguridad y recuperación de datos?	Una aplicación para tratamiento de grandes cantidades de información, donde es muy importante la efectividad de los procesos de backup y recuperación de datos.
13	Múltiples instalaciones	¿Se diseñó y desarrolló el sistema para soportar múltiples instalaciones en diferentes organizaciones?	Una aplicación software para una multinacional con oficinas en varios países.
14	Facilidad de mantenimiento	¿Se diseñó y desarrolló el sistema pensando en facilitar el posterior proceso de mantenimiento?	

Tabla 2. Las 14 características generales del sistema, para el cálculo de Puntos Función.

peticiones interactivas al sistema, ya que éstas se contabilizan por separado. Los datos de entrada son usados para mantener uno o más Ficheros Lógicos Internos (archivos maestros), siempre y cuando no representen información de control del sistema. Para determinar las Entradas Externas, se suelen examinar las pantallas de introducción de datos, los cuadros de diálogo y el formato de los formularios de entrada, si es que existen. Además, si se trata de entradas procedentes de otras aplicaciones distintas, éstas deberán necesariamente actualizar los Ficheros Lógicos Internos del sistema que se pretende medir.

- **Salidas Externas:** Cada Salida Externa es un proceso elemental a través del cual se permite la salida de datos del sistema. Estos datos suelen ser los resultados derivados de la ejecución de algoritmos o la evaluación de fórmulas, y generan informes (reports) o archivos de salida que sirven de entrada a otras aplicaciones. En la creación de estos informes o archivos de salida intervienen uno o más Ficheros Lógicos Internos o uno o más Ficheros Externos de Interfaz. Una forma de determinar las Salidas Externas de un sistema es observar los posibles informes de salida de datos y los formatos de los ficheros que sirven a otras aplicaciones

- **Consultas Externas** (o peticiones al sistema)**:** Cada Consulta Externa es un proceso elemental con componentes de entrada y de salida que consiste en la selección y recuperación de datos de uno o más Ficheros Lógicos Internos o de uno o más Ficheros Externos de Interfaz, y su posterior devolución al usuario o aplicación que los solicitó. Se trata, entonces, de peticiones interactivas que requieren una respuesta del sistema. En el proceso de entrada no se actualiza ningún Fichero Lógico Interno, y en el proceso de salida los datos devueltos no contienen datos derivados (es decir, datos resultantes de la ejecución de algoritmos o la evaluación de fórmulas). Al igual que sucedía con las Entradas Externas, una posible forma de detectar las Consultas Externas es examinando los formularios de entrada, las pantallas de entrada de datos, los cuadros de diálogo, etc...

- **Ficheros Lógicos Internos** (o archivos maestros)**:** Un Fichero Lógico Interno es un conjunto de datos definidos por el usuario y relacionados lógicamente, que residen en su totalidad dentro de la propia aplicación, y que son mantenidos a través de la Entradas Externas del sistema. Para determinar los posibles Ficheros Lógicos Internos se suelen examinar los modelos físicos y/o lógicos preliminares, los formatos de tablas, las descripciones de bases de datos, etc...

- **Ficheros Externos de Interfaz:** Un Fichero Externo de Interfaz es un conjunto de datos definidos por el usuario, que están relacionados lógicamente y que sólo son usados para propósitos de referencia. Los datos residen en su totalidad fuera de los límites de la aplicación y son mantenidos por otras aplicaciones. En definitiva, un Fichero Externo de Interfaz es un Fichero Lógico Interno para otra aplicación. Para determinar los posibles Ficheros Externos de Interfaz se suelen analizar las descripciones de interfaces del sistema con otras aplicaciones.

Agrupando las Entradas Externas, las Salidas Externas y las Consultas Externas obtendríamos el conjunto de ***Transacciones Lógicas*** del sistema, y agrupando los Ficheros Lógicos Internos y los Ficheros Externos de Interfaz obtendríamos el conjunto de ***Ficheros Lógicos*** del sistema.

Pongamos un ejemplo: en un sencillo programa de corrección ortográfica, tendríamos dos entradas (el nombre del fichero a examinar y el nombre del diccionario personalizado), tres salidas (el número total de palabras revisadas, el número de errores encontrados y la lista de palabras erróneas), dos peticiones al sistema (el usuario puede saber en cualquier instante el número de palabras procesadas hasta ese momento, y la lista de palabras erróneas), dos ficheros externos (el archivo a examinar y el diccionario personalizado) y un fichero interno (el diccionario general). Pues bien... en este ejemplo, el número total de elementos del sistema sería: 2+3+2+2+1=10.

Paso 5: Clasificación de componentes.

Una vez que se han inventariado tanto las transacciones lógicas como los ficheros lógicos, puede procederse a la clasificación de los componentes del sistema. Más exactamente, se trata de determinar la complejidad de cada componente, aunque esto es algo subjetivo. Para tratar de eliminar parte de esta subjetividad, se suelen formular preguntas concretas acerca de los componentes, como las que se muestran a continuación [IFPUG95]:

- ***Entradas Externas:***

  ¿Necesitan las Entradas Externas acceder a más o a menos de 3 ficheros?

  Para todas las Entradas Externas que referencien a más de 3 ficheros, todo lo que necesitaremos saber es si la Entrada Externa en cuestión está formada por más o a menos de 4 tipos de datos distintos. Si consta de más de 4 tipos de datos distintos, la Entrada Externa será considerada de complejidad *Alta* y si consta de menos de 4 tipos de datos distintos será considerada de complejidad *Media*. Cualquier Entrada Externa que referencie a menos de 3 ficheros se considerará de complejidad *Baja*.

- ***Salidas Externas:***

  ¿Necesitan las Salidas Externas acceder a más o a menos de 4 ficheros?

  Para todas las Salidas Externas que referencien a más de 4 ficheros, todo lo que necesitaremos saber es si la Salida Externa en cuestión está formada por más o a menos de 5 tipos de datos distintos. Si consta de más de 5 tipos de datos distintos, la Salida Externa será considerada de complejidad *Alta* y si consta de menos de 5 tipos de datos distintos será considerada de complejidad *Media*. Cualquier Salida Externa que referencie a menos de 4 ficheros se considerará de complejidad *Baja*.

- ***Consultas Externas:***

  Como cada Consulta Externa tiene un componente de entrada y otro de salida, para el componente de entrada se siguen los criterios aplicables a las Entradas Externas, y para el componente de salida los criterios aplicables a las Salidas Externas. Si se obtuvieran complejidades distintas para cada componente, desecharíamos la complejidad más baja y nos quedaríamos con la más alta.

- ***Ficheros Lógicos Internos y Ficheros Externos de Interfaz:***

  ¿Están compuestos los ficheros de un solo tipo de registro, o de más de un tipo de registro?

  Si todos o la mayoría de los ficheros contienen un solo tipo de registro, entonces todo lo que necesitamos saber es si el fichero contiene más o menos de 50 tipos de datos distintos. Si contiene más de 50 tipos de datos distintos, el fichero será considerado de complejidad *Alta* y si contiene menos de 50 tipos de datos distintos será considerado de complejidad *Baja*. Cualquier fichero que esté formado por más de un tipo de registro, deberá considerarse aparte y ser contado por separado.

Para Ficheros Lógicos Internos y Ficheros Externos de Interfaz				**Para Salidas Externas y Consultas Externas**				**Para Entradas Externas**			
Nº de tipos de registro	**Tipos de Datos distintos**			**Nº de ficheros referenc**	**Tipos de Datos distintos**			**Nº de ficheros referenc**	**Tipos de Datos distintos**		
	1-19	**20-50**	**+51**		**1-5**	**6-19**	**+20**		**1-4**	**5-15**	**+16**
1	Baja	Baja	Media	**0 - 1**	Baja	Baja	Media	**0 - 1**	Baja	Baja	Media
2 - 5	Baja	Media	Alta	**2 – 3**	Baja	Media	Alta	**2 – 3**	Baja	Media	Alta
+6	Media	Alta	Alta	**+4**	Media	Alta	Alta	**+3**	Media	Alta	Alta

Tabla 3. Clasificación de los componentes del sistema, según el modelo COCOMO II.

Una clasificación más precisa y exhaustiva nos la proporciona el modelo de estimación COCOMO II, comentado en el *Apartado 6* de este trabajo. Según este modelo, la complejidad de los componentes del sistema para el conteo de Puntos Función se determina según la tabla 3 mostrada en la página anterior.

Una vez que hemos clasificado los componentes y determinado su complejidad, hemos de saber qué valor en Puntos Función hemos de asignarle a cada uno de ellos. En la tabla 4 del Paso 7 se indican estos valores.

Paso 6: Revisión de las 14 características generales del sistema (GSC's).

Para asegurar una mayor precisión en los resultados, es necesario volver a calcular el Factor de Ajuste (VAF) tal y como lo hicimos en el paso 3, es decir, contestando a las 14 preguntas de la tabla 2 y utilizando la fórmula:

$$VAF = 0.65 + 0.01\sum_{i=1}^{14} F_i$$

Es muy importante determinar un valor preciso para el VAF, ya que influye en un ±35% en la cuenta total de Puntos Función.

Paso 7: Tabulación de resultados.

Para mostrar los resultados obtenidos en los pasos precedentes, puede construirse una tabla como la que se muestra a continuación, donde se indican los tipos de componentes contabilizados (en filas) y la complejidad asociada a cada uno de ellos (en columnas). Comentaremos los pasos principales a seguir, hasta obtener el valor total en Puntos Función del sistema que se pretende medir:

1. Multiplicar cada tipo de componente por el valor en Puntos Función indicado, dependiendo de su complejidad.
2. Sumar por filas los resultados obtenidos en el punto 1. Se obtiene así el un valor total en Puntos Función para cada tipo de componente.
3. Sumar todos los valores de la columna *'Total'*. Se obtiene así el ***Número Total de Puntos Función sin Ajustar***, que responde a la siguiente expresión:

$$PFsA = \sum_{i=1}^{15} (n^{o}\ de\ componentes\ de\ tipo\ i * peso\ del\ componente\ i)$$

   Como podemos observar tendríamos, en teoría, 15 componentes distintos (3 niveles de complejidad, por 5 tipos distintos de componentes), con lo que el valor en *Puntos Función sin Ajustar* (*PFsA*) será la suma de cada componente del sistema, por su peso.
4. Calcular el Factor de Ajuste (VAF), también llamado Factor de Complejidad Técnica (referirse al *Paso 3: Cálculo del Factor de Ajuste*).
5. Multiplicar el VAF calculado en el punto anterior, por el valor de *Puntos Función sin Ajustar* obtenido en el punto 3. Se obtiene así el ***Número Total de Puntos Función Ajustados*** (*PFA*), siendo éste el resultado total de la cuenta.

$$PFA = VAF * PFsA$$

Componente	Complejidad del componente (factor de peso)			Total
	Baja	Media	Alta	
Entradas Externas	____ x 3 = ____	____ x 4 = ____	____ x 6 = ____	____
Salidas Externas	____ x 4 = ____	____ x 5 = ____	____ x 7 = ____	____
Consultas Externas	____ x 3 = ____	____ x 4 = ____	____ x 6 = ____	____
Ficheros Lógicos Internos	____ x 7 = ____	____ x 10 = ____	____ x 15 = ____	____
Ficheros Externos de Interfaz	____ x 5 = ____	____ x 7 = ____	____ x 10 = ____	____
			Nº Total de Puntos Función sin Ajustar (PFsA):	____
			Factor de Ajuste (VAF):	x ___
			Nº Total de Puntos Función Ajustados (PFA):	_____

Tabla 4. Conteo de Puntos Función.

Paso 8: Validación de resultados.

Antes de utilizar el resultado final obtenido para posibles estimaciones, el proceso de conteo descrito en los pasos anteriores deberá ser exhaustivamente revisado para detectar posibles errores u omisiones que hayan podido producirse. Deberemos verificar la completitud del Análisis de Puntos Función llevado a cabo, y asegurarnos de que se han tenido en cuenta todos los componentes del sistema, sin obviar ninguno.

Para ayudar en este proceso de validación de resultados y, sobre todo, para hacernos una idea de la validez del proceso seguido y de los resultados obtenidos, existen una serie de preguntas que cabría formularnos. Son las siguientes:

1. ¿Se ha planificado el conteo de Puntos Función como una tarea más del proceso de desarrollo?
2. La persona que ha llevado a cabo el conteo de Puntos Función, ¿ha sido previamente entrenada? ¿Posee el certificado correspondiente que acredite sus conocimientos en FPA?
3. ¿Se ha utilizado la documentación apropiada para el proceso de conteo?
4. ¿Se siguieron los criterios y reglas propuestos por la IFPUG en su *"Counting Practices Manual"*?
5. El proceso de conteo, ¿se hizo siempre desde el punto de vista del usuario, de sus requisitos y necesidades?
6. El proceso de conteo, ¿se hizo siempre desde un punto de vista lógico del sistema, e independientemente de cualquier implementación física?
7. El número de componentes que se han detectado en el sistema a medir, ¿está acorde con el número medio de componentes observados en otros sistemas? (20% en Entradas Externas, 25% en Salidas Externas, 10% en Consultas Externas, 40% en Ficheros Lógicos Internos, y 5% en Ficheros Externos de Interfaz). Si no es así, ¿existe una razón lógica que lo explique?
8. Cuando se realizó el inventariado de transacciones lógicas y ficheros lógicos (paso 4), ¿se revisó exhaustivamente para detectar posibles errores u omisiones?

9. ¿Se ha comparado el valor final obtenido con otras mediciones de proyectos de similar envergadura? ¿Se guardará dicho valor para utilizarlo en futuras comparaciones con otros proyectos?
10. El resultado final obtenido, ¿ha sido revisado por otros expertos en conteo de Puntos Función?
11. etc, etc,...

3.4. El Modelo Funcional y su papel en el Análisis de Puntos Función:

Como sabemos, el Análisis de Puntos Función (FPA) es aplicable desde las primeras etapas del ciclo de vida de un proyecto, más concretamente desde el momento en que queda perfectamente determinada la funcionalidad del software. El resultado de su aplicación sobre un proyecto software es la medida de su tamaño funcional en términos de Puntos Función. Además, como veremos más adelante (punto 3.5), el uso del FPA puede extenderse fácilmente a la estimación de otros parámetros, como el esfuerzo de desarrollo y la duración del proyecto, siempre en conjunción con técnicas de Macro-Estimación. Puede decirse, entonces, que los Puntos Función son sólo el numerador o el denominador de muchas otras mediciones.

Pero la aplicación del FPA a un proyecto software requiere la existencia de un *Modelo Funcional* del sistema, sobre el cual apoyarnos firmemente para la aplicación del método. La descripción de este Modelo Funcional es lo que trataremos en los siguientes apartados.

3.4.1. Componentes del Modelo Funcional:

El modelo funcional trata de representar el sistema software tal y como lo vería el usuario. Se trata, en esencia, de una vista externa del sistema que muestra la funcionalidad que proporciona el software a los usuarios.

El término *"usuario"* hace referencia no sólo a los usuarios finales de la aplicación, sino también a cualquier otra entidad que interactúe con el sistema: el personal de soporte técnico, el administrador del sistema, otras aplicaciones software que se comuniquen con nuestra aplicación, enviando o recibiendo datos de ella, etc...

El modelo funcional consta de seis componentes básicos, que son los siguientes:

- Transacciones lógicas.
- Ficheros lógicos.
- Interrelaciones entre las transacciones lógicas.
- Interrelaciones entre los ficheros lógicos y las transacciones lógicas.
- Procedimiento para indicar el impacto provocado por cambios en el software.
- Atributos asignados a transacciones lógicas.

Cada uno de estos componentes se describe a continuación:

Transacciones lógicas:

Una transacción lógica representa una interacción del usuario con el sistema software. Es lo que el usuario percibe como una *"unidad de trabajo"* o *"tarea"* a realizar por el sistema.

Poseen las siguientes características:

- Se definen y describen en un lenguaje comprensible por el usuario final (p. ej.: pseudocódigo, lenguaje natural, etc...).
- Es uno de los componentes del sistema software que se tiene en cuenta para el conteo de Puntos Función.
- Hacen posible la comunicación interactiva entre el sistema y el usuario, permitiendo a este último tanto la introducción de datos al sistema, como la modificación y recuperación de los mismos.

Si suponemos un sistema software para el control de compras y ventas de una empresa genérica, algunas posibles transacciones lógicas podrían ser: *Añadir proveedor, Añadir cliente, Borrar Proveedor, Modificar datos de cliente, Imprimir factura, Pagar a proveedor, Obtener ventas por ciudades,* etc... Es lo que el estándar ISO 14143 para Medida del Tamaño Funcinal llama BFC's (Componentes Funcionales Básicos) (ver *Apartado 9*).

Ficheros lógicos.

Un fichero lógico es un almacén de datos. Representa a un conjunto de datos almacenados y perfectamente conocidos por el usuario, ya que éste puede acceder directamente a ellos a través de transacciones lógicas de entrada, modificación o recuperación de dichos datos.

Al igual que sucedía con las transacciones lógicas, los ficheros lógicos se definen y describen en un lenguaje comprensible por el usuario y son también un componente fundamental del sistema software, sobre el que recae el conteo de Puntos Función.

Siguiendo con el ejemplo propuesto en el punto anterior, algunos posibles ficheros lógicos serían: *Datos de proveedores, Datos de clientes, Datos de materias primas, Datos de productos elaborados,* etc...

Interrelaciones entre transacciones lógicas.

Las transacciones lógicas se organizan en torno a una jerarquía de la que pueden deducirse interrelaciones entre unas transacciones y otras (ver ejemplo en figura 3). El último nivel de esta jerarquía está compuesto por las propias transacciones lógicas, pudiendo estar agrupadas en función de algún criterio concreto; por ejemplo, según el fichero lógico al cual acceden. Siguiendo con el supuesto anterior, una posible agrupación de transacciones podría ser: *Añadir cliente, Modificar datos de cliente, Borrar cliente* y *Listar clientes*. Como vemos, todas estas *'tareas'* accederían al mismo fichero de datos: al fichero de *Clientes*.

En los niveles superiores de la jerarquía tendríamos los bloques funcionales o subsistemas del sistema software, mientras que en los niveles intermedios estarían las actividades principales de cada bloque funcional. Según esto, una transacción lógica podría quedar completamente definida de la siguiente manera:

Gestión de compras y ventas / Gestionar ventas / Gestionar pedido cliente / Servir pedido.

...donde *'Gestión de compras y ventas'* sería uno de los bloques funcionales del programa y *'Servir pedido'* sería una de las posibles transacciones lógicas de la actividad *'Gestionar pedido cliente'*.

Esta clasificación jerárquica nos permitirá, más adelante, calcular el tamaño funcional de cada subsistema o bloque funcional, es decir, de las transacciones lógicas que pertenecen a cada una de las ramas de la jerarquía. De esta manera podremos establecer comparaciones entre los tamaños funcionales de los distintos subsistemas que componen la aplicación software.

Interrelaciones entre los ficheros lógicos y las transacciones lógicas.

Para poder llevar a cabo su función, cada *tarea* o transacción lógica utiliza datos almacenados en los ficheros lógicos del sistema. Entonces, estos ficheros lógicos están de alguna forma enlazados o relacionados con las transacciones lógicas que hacen uso de ellos, de tal manera que se puede llevar un cierto control de cuándo una transacción lee o actualiza datos del fichero.

Además, esta interrelación transacción-fichero permite asegurar que el modelo funcional es *completo,* es decir, que todas las transacciones lógicas tienen sus correspondientes ficheros lógicos a los que acceden, y que todos los ficheros lógicos son accedidos por acciones del usuario, a través de la *tarea* o transacción lógica asociada.

Procedimiento para indicar el impacto provocado por cambios en el software.

Como sabemos, un proyecto de desarrollo software es un conjunto de actividades, a cuyo término se obtendrá como resultado una nueva aplicación software. El alcance de la misma se corresponderá (o deberá corresponderse) con las tareas o transacciones lógicas definidas en las primeras etapas del ciclo de desarrollo.

Por el contrario, un proyecto de *mejora* de una aplicación software existente, realizará cambios en la funcionalidad de dicha aplicación. En definitiva, añadirá, modificará o eliminará funcionalidad del sistema. Y el alcance de todas estas modificaciones se define en el modelo funcional a través de:

- Una extensión del modelo para contemplar la nueva funcionalidad añadida.
- Marcado del componente funcional afectado, con la etiqueta apropiada: *'modificado'*, *'eliminado'*, *'añadido'...*

Atributos asignados a transacciones lógicas.

Es posible asignar ciertos atributos a las tareas o transacciones lógicas definidas en el modelo, de tal forma que proporcionen información concreta sobre ciertas partes del sistema. Esto facilita tanto el análisis del sistema en su conjunto, como el análisis a nivel de sus componentes o módulos funcionales. Por ejemplo, podrá calcularse el tamaño funcional de una parte del sistema, considerando únicamente las transacciones lógicas que tengan el mismo valor para un determinado atributo (p. ej.: *prioridad de terminación*).

La siguiente tabla muestra algunos ejemplos de los atributos comentados, junto con posibles valores que podrían tomar:

Atributo:	Posibles valores:
Plataforma	VAX, PC (DOS), PC (OS/2),...
Lenguaje	C, C++, PowerBuilder, Cobol,...
Origen	Desarrollado, Comprado, Comprado y adaptado,...
Versión	Versión 1.0, Versión 2.0,...
Estado	No comenzado, En progreso, Terminado,...
Prioridad de terminación	Alta, Media, Baja,...

Tabla 5. Atributos asignados a transacciones, y posibles valores.

3.4.2. Representación del modelo funcional:

En el proceso de creación del modelo funcional de un sistema, una vez que se han determinado sus seis componentes principales, es necesario representarlo de alguna forma. Habitualmente, dicha representación se hace en forma de jerarquía funcional, mostrando tanto las transacciones lógicas existentes como las interrelaciones entre ellas. Además, con esta representación se muestra toda la funcionalidad que el sistema implementa para el usuario.

La jerarquía funcional comentada puede expresarse bien de forma gráfica (véase ejemplo en figura 3), o bien de forma textual (véase ejemplo en figura 4). Pero también existen otras representaciones del modelo funcional. Por ejemplo:

- A través de una lista de transacciones lógicas y sus ficheros lógicos asociados.
- A través de una lista detallada para cada transacción lógica, mostrando sus interrelaciones con los ficheros lógicos.
- A través de una lista detallada para cada fichero lógico, mostrando sus interrelaciones con las transacciones lógicas.

3.4.3. Extensión del modelo funcional para su medida en Puntos Función:

En el FPA, a cada transacción lógica y a cada fichero lógico se le asocia un valor determinado en Puntos Función, según las reglas establecidas por la IFPUG en su *"Counting Practices Manual"* [IFPUG95], y que pasamos a resumir a continuación:

- A cada transacción lógica se le asigna un valor, en función de su tipo (*Entrada Externa, Salida Externa* o *Consulta Externa*) y complejidad (*Baja, Media* o *Alta*). Esta última se determina de forma objetiva a partir de una serie de reglas establecidas por la IFPUG, que se basan en los tipos de datos con los que se trabaja y en los ficheros lógicos a los que se accede.
- A cada fichero lógico se le asigna un valor, en función de su tipo (*Fichero Lógico Interno* o *Fichero Externo de Interfaz*) y su complejidad (*Baja, Media* o *Alta*). Al igual que antes, la complejidad de los ficheros lógicos se determina objetivamente a partir de reglas establecidas por la IFPUG, las cuales se basan en los tipos de datos utilizados y en los tipos de registro del fichero.

Los valores concretos que se asignan tanto a las transacciones lógicas como a los ficheros lógicos pueden consultarse en la tabla 4.

Una vez que hemos asignado los valores a cada uno de los componentes, obtendremos un resultado parcial de la medición si sumamos todos ellos, de forma independiente para transacciones lógicas y para ficheros lógicos. El valor que obtenemos es aún un valor *no ajustado*, es decir, falta determinar el factor de ajuste (VAF) por el que se ha de multiplicar el valor no ajustado. Una vez más, el VAF se calcula según las reglas establecidas por la IFPUG, basándose en un total de 14 preguntas acerca del sistema (referirse al paso 3 de la sección 3.3, y consultar la tabla 2). El resultado obtenido es ya el tamaño funcional *ajustado* del conjunto de transacciones lógicas o conjunto de ficheros lógicos que se está midiendo.

Por ejemplo: la tabla que sigue muestra un supuesto en el que se mide el tamaño funcional de todos los ficheros lógicos de un determinado sistema. Notar como para cada fichero lógico se determina su tipo, su complejidad y el valor en Puntos Función que le corresponde:

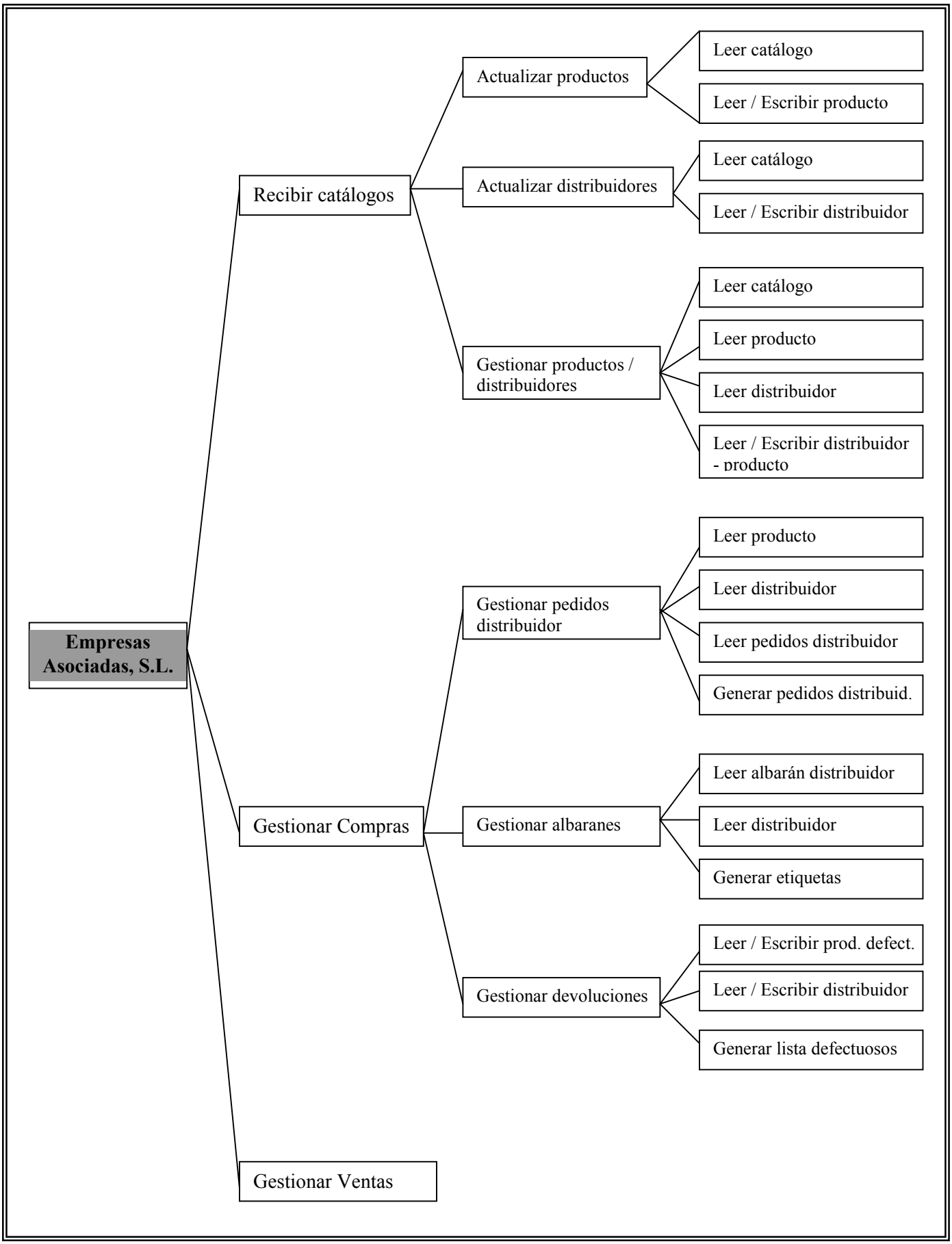

Figura 3. Representación gráfica de un modelo funcional.

Empresas Asociadas, S.L.

1. Recibir Catálogos ⇒
   - **1.1. Actualizar productos ⇒**
     - 1.1.1. Leer catálogo.
     - 1.1.2. Leer / Escribir producto.
   - **1.2. Actualizar distribuidores ⇒**
     - 1.2.1. Leer catálogo.
     - 1.2.2. Leer / Escribir distribuidor.
   - **1.3. Gestionar productos / distribuidores ⇒**
     - 1.3.1. Leer catálogo.
     - 1.3.2. Leer producto.
     - 1.3.3. Leer distribuidor.
     - 1.3.4. Leer / Escribir distribuidor-producto.

2. Gestionar Compras ⇒
   - **2.1. Gestionar pedidos distribuidor ⇒**
     - 2.1.1. Leer producto.
     - 2.1.2. Leer distribuidor.
     - 2.1.3. Leer pedidos distribuidor.
     - 2.1.4. Generar pedidos distribuidor.
   - **2.2. Gestionar albaranes ⇒**
     - 2.2.1. Leer albarán distribuidor.
     - 2.2.2. Leer distribuidor.
     - 2.2.3. Generar etiquetas.
   - **2.3. Gestionar devoluciones ⇒**
     - 2.3.1. Leer / Escribir producto defectuoso.
     - 2.3.2. Leer / Escribir distribuidor.
     - 2.3.3. Generar lista defectuosos.

3. Gestionar Ventas ⇒
   - **3.1. Gestionar presupuesto cliente ⇒**
     - 3.1.1. Leer solicitud de presupuesto.
     - 3.1.2. Leer cliente.
     - 3.1.3. Leer producto.
     - 3.1.4. Leer / Escribir presupuesto.
     - 3.1.5. Generar presupuesto.
   - **3.2. Gestionar pedido cliente ⇒**
     - 3.2.1. Generar pedidos.
     - 3.2.2. Gestionar pedidos completos.
     - 3.2.3. Gestionar pedidos no completos.
     - 3.2.4. Servir pedidos.
   - **3.3. Gestionar facturas cliente ⇒**
     - 3.3.1. Leer albarán cliente.
     - 3.3.2. Leer cliente.
     - 3.3.3. Leer producto.
     - 3.3.4. Generar factura cliente.
   - **3.4. Gestionar devolución cliente ⇒**
     - 3.4.1. Leer devolución cliente.
     - 3.4.2. Leer cliente.
     - 3.4.3. Escribir albarán cliente.
     - 3.4.4. Escribir producto defectuoso.
   - **3.5. Gestionar clientes ⇒**
     - 3.5.1. Leer datos cliente.
     - 3.5.2. Leer / Escribir cliente.

Figura 4. Descripción textual de un modelo funcional.

Ficheros del sistema				
Identificador	**Descripción**	**Tipo**	**Complejidad**	**Puntos Función**
MATEPRIM	Datos de Materias Primas	Interfaz Externo	Baja	5
PRODUCT	Datos de Productos Elaborados	Interno	Baja	7
PROVEED	Datos de Proveedores	Interfaz Externo	Baja	5
CLIENTE	Datos de Clientes	Interfaz Externo	Baja	5
TRANSPORT	Datos de Transportistas	Interfaz Externo	Baja	5
CAMION	Datos de Camiones	Interno	Baja	7
PORTES	Datos sobre Portes	Interno	Baja	7
		Nº de Archivos:	**7**	
		Puntos Función sin Ajustar:		**41**
		Factor de Ajuste (FAV):		**1.08**
		Puntos Función Ajustados:		**44**

Tabla 6. Características relevantes del conjunto de ficheros lógicos de un sistema.

Sumando los Puntos Función obtenidos para las transacciones lógicas, con los obtenidos para los ficheros lógicos, se obtiene el tamaño funcional de la aplicación que se está midiendo.

Una posible manera de representar el tamaño funcional obtenido es desglosándolo según el tipo de componente y la complejidad del mismo, tal y como se muestra en el ejemplo de la siguiente tabla:

Tamaño funcional del sistema				
Tipo Componente	**Complejidad**	**Número de Componentes**	**Puntos Función Asignados**	**Puntos Función Totales**
Entradas Externas	Baja	21	3	63
	Media	1	4	4
	Alta	0	6	0
	Subtotal:	**22**		**67**
Salidas Externas	Baja	2	4	8
	Media	5	5	25
	Alta	3	7	21
	Subtotal:	**10**		**54**
Consultas Externas	Baja	6	3	18
	Media	3	4	12
	Alta	0	6	0
	Subtotal:	**9**		**30**
Ficheros Lógicos Internos	Baja	4	7	28
	Media	0	10	0
	Alta	0	15	0
	Subtotal:	**4**		**28**
Ficheros Externos de Interfaz	Baja	4	5	20
	Media	0	7	0
	Alta	0	10	0
	Subtotal:	**4**		**20**
	Nº Total de Puntos Función sin Ajustar:			**199**
	Factor de Ajuste (VAF):			**1.08**
	Nº Total de Puntos Función Ajustados (≡Tamaño Funcional)**:**			**215**

Tabla 7. Tamaño funcional de una aplicación, desglosado según tipo de componente y complejidad del mismo.

3.5. Aplicación del tamaño funcional en la estimación de proyectos software:

Como sabemos, el proceso de desarrollo de aplicaciones software, además de costoso, es también bastante complejo. Y esa complejidad hace que sean realmente difíciles de predecir o estimar aspectos tan relevantes como el esfuerzo de desarrollo o el tamaño final de la aplicación, más aún si se hace en las primeras etapas del ciclo de vida. Por este hecho, es importante disponer de un buen método de estimación capaz de obtener buenos resultados durante esas primeras etapas, cuando los datos disponibles sobre la aplicación son aún muy escasos.

En este ámbito de aplicación, el Análisis de Puntos Función (FPA) tiene un papel muy importante que desarrollar, convirtiéndose en uno de los métodos de estimación más importantes y más ampliamente utilizado en todo el mundo.

3.5.1. Estimación del esfuerzo de desarrollo:

La estimación del esfuerzo de desarrollo de un proyecto software ha de basarse en los datos del proyecto que se conozcan en el momento de la medición. Hay muchos factores que influyen en la productividad del proceso de desarrollo: requisitos funcionales de la aplicación, planificación de tareas, calidad y coste del producto a desarrollar, entorno de trabajo, nivel de preparación del equipo de desarrollo, tecnologías, métodos, etc...

Básicamente, hay dos técnicas generales de estimación del esfuerzo que son, quizás, las más utilizadas hoy en día:

- **Micro-Estimación:** Consiste en estimar por separado el esfuerzo de desarrollo asociado a cada componente o actividad que deba implementar el software. El resultado final de la estimación viene dado por la suma de todos los esfuerzos obtenidos. Se trata de una aproximación al resultado *"de abajo arriba"* (bottom-up).

- **Macro-Estimación:** Consiste en extrapolar al nuevo proyecto los resultados obtenidos de estimaciones realizadas sobre otros ya desarrollados, siempre y cuando éstos tengan características similares. Es decir, se hace uso de la experiencia observada en proyectos anteriores. Se trata de una aproximación al resultado *"de arriba abajo"* (top-down).
  Para evitar que la extrapolación de resultados se realice simplemente por analogía entre proyectos, muy a menudo se utilizan uno o varios algoritmos que realizan una estimación del esfuerzo en función de un número de variables consideradas *clave.*

Cada una de estas técnicas tiene sus puntos fuertes y sus puntos débiles, según se muestra en la siguiente tabla:

Técnica	Pros	Contras
Micro-Estimación	Detallado.	Subjetivo, tiende a ser optimista. Puede pasar por alto componentes o actividades del software.
Macro-Estimación	Basado en la experiencia. Permite la utilización de algoritmos. Objetivo, verificable, eficiente.	Basado en la experiencia anterior. Para obtener mejores resultados, precisa ser reajustado según el entorno, la organización, etc...

Tabla 8. Pros y contras de las técnicas de estimación del esfuerzo.

En general, el uso de la técnica de Macro-Estimación no es muy útil cuando se trata de proyectos muy pequeños[1]. Según afirma Capers Jones[2] [JON96] en su artículo "Applied Software Measurement": *"Los proyectos muy pequeños varían tanto* (en lo que a productividad se refiere) *que no suelen ser siempre útiles para propósitos de investigación tecnológica"*.

En algunas ocasiones la técnica de Macro-Estimación puede funcionar de forma parecida a la Micro-Estimación, es decir, obteniendo el resultado final como suma de estimaciones parciales aplicadas sobre distintos bloques funcionales del software. Estos casos son habituales cuando:

1. El proyecto objeto de la estimación mezcla diferentes niveles tecnológicos. Por ejemplo: el generador de informes de una aplicación puede estar implementado en un lenguaje de cuarta generación y el resto del programa en otro lenguaje de nivel inferior.
2. El proyecto presenta bloques funcionales con cierta complejidad técnica, algorítmica, etc... El ratio de productividad esperado será menor que el de otros bloques funcionales no tan complejos.
3. El proyecto presenta bloques funcionales reutilizados. Es posible que ciertos bloques de la aplicación hayan sido tomados de proyectos ya concluidos, o incluso hayan sido comprados a terceros, como sucede con los módulos de seguridad, herramientas de copia, búsqueda y recuperación de datos, etc... En estos casos el ratio de productividad obtenido dependerá de la cantidad de modificaciones que haya que hacer sobre dichos módulos para integrarlos en el proyecto y generalmente este ratio suele ser mayor que si optamos por desarrollar nosotros mismos el componente.

3.5.2. Utilización de los Puntos Función en la estimación del esfuerzo:

La tabla que se muestra a continuación pone de manifiesto cómo el Análisis de Puntos Función (FPA) puede utilizarse en la estimación del esfuerzo de desarrollo, en conjunción con las dos técnicas vistas anteriormente:

Técnica	Uso de los Puntos Función
Micro-Estimación	El alcance del proyecto, definido en el primer paso del Análisis de Puntos Función, ayuda a determinar cuáles son las tareas a realizar.
Macro-Estimación	El tamaño funcional permite calcular la productividad 'esperada', según lo observado en proyectos anteriores. El tamaño funcional es un dato de entrada clave para la mayoría de los algoritmos de estimación.

Tabla 9. Utilización del FPA en la estimación del esfuerzo de desarrollo.

- **Estimación Indicativa o 'Ball-park':**

La llamada "Estimación Indicativa" o "Ball-park" es la técnica de Macro-Estimación que se utiliza habitualmente en situaciones de falta de información sobre el proyecto. Capers Jones propone la siguiente ecuación para determinar el esfuerzo de desarrollo de un proyecto:

$$Esfuerzo = \left(\frac{Tamaño\ en\ PF}{150} \right) * Tamaño\ en\ PF^{0'4}$$

[1] Los expertos en métricas del software establecen el tamaño máximo de una aplicación pequeña en torno a los 50-100 Puntos Función, si bien es cierto que no existe consenso general con respecto a este tema.

[2] Capers Jones es vicepresidente ejecutivo de Artemis Management Systems, fundador de la compañía Software Productivity Research, técnico de calidad del software en IBM durante 12 años y miembro de la IEEE y de la IFPUG.

Por ejemplo, para un proyecto con 1000 Puntos Función...

$$Esfuerzo = \left(\frac{1000}{150}\right) * 1000^{0'4} = 105'5 \ \ meses \ \ de \ \ trabajo$$

Estos 105'5 meses de trabajo suponen unas 14770 horas de desarrollo, suponiendo una jornada laboral de 35 horas semanales. Es decir, una única persona trabajando en el desarrollo del proyecto debería invertir 14770 horas hasta su finalización, por lo que si suponemos un equipo de desarrollo de 10 personas, cada una de ellas debería invertir 1477 horas de trabajo. De esta forma, en 10'5 meses habría concluido el proyecto.

Otras ecuaciones que conviene considerar son las propuestas por la ISBSG (International Software Benchmarking Standards Group). Son las siguientes:

Para todos los proyectos:	*Esfuerzo = 11'79 * (Tamaño en PF)$^{0'898}$*
Para proyectos en 3GL[3]:	*Esfuerzo = 5'76 * (Tamaño en PF)$^{1'062}$*
Para proyectos en 4GL[4]:	*Esfuerzo = 9'32 * (Tamaño en PF)$^{0'912}$*

Por ejemplo, para un proyecto con 1000 Puntos Función...

Para todos los proyectos:	*Esfuerzo = 11'79 * 1000$^{0'898}$ = 5828 horas de trabajo = 5'8 horas/PF*
Para proyectos en 3GL:	*Esfuerzo = 5'76 * 1000$^{1'062}$ = 8839 horas de trabajo = 8'8 horas/PF*
Para proyectos en 4GL:	*Esfuerzo = 9'32 * 1000$^{0'912}$ = 5075 horas de trabajo = 5'1 horas/PF*

- **Estimación Consolidada:**

Para una estimación del esfuerzo de desarrollo más precisa y fiable, se suele utilizar la técnica de Micro-Estimación aplicada sobre las tareas o componentes funcionales de la aplicación. Además, se suele utilizar la Macro-Estimación para validar los resultados obtenidos en la Micro-Estimación, de tal forma que si de una técnica a otra los resultados varían más de un 10-15%, entonces será conveniente volver a repetir el proceso de estimación.

El tamaño funcional es sólo uno de los muchos factores que influyen en el esfuerzo de desarrollo de una aplicación pero, sin embargo, es considerado como *factor clave*: según aumenta el tamaño funcional, así lo hace el esfuerzo de desarrollo requerido. Pero esta correlación no es lineal, sino que el esfuerzo asociado crece más rápidamente que el tamaño funcional. De ahí se deduce que a medida que crece el tamaño del proyecto, decrece la productividad en el proceso de desarrollo.

La relación comentada entre esfuerzo y tamaño funcional queda reflejada en la siguiente ecuación:

*Esfuerzo = Tamaño en PF * Ratio de Productividad*

...donde *Ratio de productividad* viene expresado en *Horas/PF* o *PF/Mes*, y se obtiene habitualmente de la experiencia anterior con proyectos de similares características. Pero... ¿Cuándo dos proyectos tienen características en común? ¿Cuándo puede decirse que son similares? Para eliminar parte de la subjetividad que entrañan estas preguntas, se definen a continuación un conjunto de atributos por los que debemos regirnos a la hora de comparar dos proyectos:

[3] 3GL: Lenguajes de tercera generación.
[4] 4GL: Lenguajes de cuarta generación.

Atributo	Se refiere a...
Tipo de proyecto	Desarrollo, mejora, reingeniería, etc...
Tamaño	Tamaño funcional (en Puntos Función).
Objetivos o metas	En cuanto a calidad, coste, planificación...
Plataforma de desarrollo	Mainframe, PC...
Lenguaje utilizado	Lenguaje o generación a la que pertenece (3GL, 4GL...).
Tareas a realizar	Proyectos similares en lo que se refiere a actividades a desarrollar, *entregables* de cada actividad, etc...

Tabla 10. Atributos para comparación de proyectos.

En el proceso de búsqueda de proyectos que se asemejen a aquél que estamos desarrollando, se hace casi imprescindible la utilización de alguna herramienta software de estimación que simplifique dicho proceso. Pero también muchas veces es necesario utilizar el propio *sentido común*.

Por ejemplo: Supongamos que en un proyecto anterior se aplicó por vez primera un nuevo método o una nueva tecnología. Su ratio de productividad asociado se vio negativamente afectado por el desconcierto y el desconocimiento que supuso su utilización. Pues bien... si resulta que éste es el proyecto *'más parecido'* al que estamos desarrollando y, además, vamos a utilizar también el mismo método o tecnología, entonces se espera que el ratio de productividad del nuevo proyecto sea superior al del proyecto anterior, puesto que ya habremos adquirido la experiencia que supuso la utilización del nuevo método, es decir, ya no existirá ese desconcierto y desconocimiento sobre el mismo.

Otro ejemplo: Si un proyecto es similar a otro ya desarrollado, pero con la salvedad de que en el nuevo hay que añadir alguna funcionalidad adicional, se espera que el ratio de productividad de éste sea inferior.

Como se comentó anteriormente, muchas veces es necesario disponer de un *ratio de productividad* de referencia, obtenido de la experiencia anterior con otros proyectos similares. A este respecto, cada organización suele disponer de una base de datos histórica con información relevante de cada una de las aplicaciones que ha desarrollado. Estos datos pueden ser útiles y representativos cuando se utilizan a nivel interno por la propia organización, como referencia para nuevos proyectos que haya que abordar. Sin embargo, sin han de servir de referencia a otras organizaciones, dichos datos pueden perder valor.

Esto es debido a que cada organización tiene características propias que influyen tanto en sus procesos de desarrollo como en la productividad de dichos procesos. Algunas de estas características son difíciles de identificar, cuanto más de cuantificar: ambiente de trabajo, estructura de la organización, relaciones con los clientes/usuarios, nivel de preparación del equipo de desarrollo, etc... Todos estos factores van implícitos en los ratios de productividad que se derivan, y puede ser que no sean lo suficientemente representativos para otras organizaciones.

Entonces, la solución propuesta es recurrir a bases de datos alternativas, las cuales guardan información más objetiva sobre los proyectos (tamaño, esfuerzo de desarrollo, defectos encontrados, tecnologías utilizadas...) y no dejan influirse por características más subjetivas, propias de cada organización. La ISBSG mantiene una base de datos de estas características.

3.5.3. Estimación de la duración del proyecto:

Conociendo el tamaño funcional de una aplicación software en las primeras etapas del ciclo de vida, puede estimarse la duración total del proceso de desarrollo. Para ello, Capers Jones propone la siguiente ecuación:

$$Duración = (Tamaño\ en\ PF)^{0'4}$$

...obteniéndose la duración del proyecto en meses.

Por ejemplo, para un proyecto de 1000 Puntos Función, tenemos: $Duración = 1000^{0'4} = 15'85\ meses$.

Por su parte, la ISBSG propone ecuaciones específicas según el lenguaje de desarrollo utilizado:

Para todos los proyectos:	$Duración = 0'80 * (Tamaño\ en\ PF)^{0'404}$
Para proyectos en 3GL:	$Duración = 0'33 * (Tamaño\ en\ PF)^{0'559}$
Para proyectos en 4GL:	$Duración = 1'11 * (Tamaño\ en\ PF)^{0'342}$

Por ejemplo, para un proyecto con 1000 Puntos Función...

Para todos los proyectos:	$Duración = 0'80 * 1000^{0'404} = 13'03\ meses$
Para proyectos en 3GL:	$Duración = 0'33 * 1000^{0'559} = 15'68\ meses$
Para proyectos en 4GL:	$Duración = 1'11 * 1000^{0'342} = 11'78\ meses$

Por último, Oligny, Bourque y Abran (1997) [Serge *et al*, 1997], en estudios llevados a cabo recientemente, proponen la siguiente ecuación que relaciona la duración del proyecto con el esfuerzo de desarrollo:

$$Duración = 0'662 * (Esfuerzo)^{0'328}$$

3.6. Auditoría del conteo de Puntos Función:

Los términos *auditor* y *auditoría* nos hacen sentir intranquilos a muchos de nosotros. Muchas industrias tienen inspectores y auditores independientes. En la medida en que los Puntos Función se vuelven más ampliamente usados y se convierten en una parte importante para la toma de decisiones, aquellos que utilizan el conteo de Puntos Función querrán que sean independientemente revisados y auditados.

La *auditoría* es el proceso mediante el cual una persona competente e independiente acumula y evalúa evidencias acerca de información cuantificable relacionada con una entidad específica, para el propósito de determinar y reportar el grado de correspondencia entre la información cuantificable y los criterios establecidos.

Un auditor de Puntos Función debe ser independiente, aunque puede ser también un miembro de la propia compañía (quizás un miembro del equipo de métricas).

Para realizar una auditoría de cualquier tipo, debe haber información verificable y que cumpla con un estándar, para que el auditor pueda evaluar dicha información.

3.6.1 Acumulación y Evaluación de Evidencias:

La principal tarea de una *auditoría de cumplimiento* es determinar si un conteo de Puntos Función sigue los procedimientos y guías establecidos por el comité de prácticas de conteo del grupo IFPUG. Los resultados de una *auditoría de cumplimiento* son generalmente entregados a alguien dentro de la organización que está siendo auditada, en vez de hacerlos públicos y darlos a conocer a los usuarios.

La ***evidencia*** se define como cualquier información usada por el auditor, para determinar si el conteo de los Puntos Función que está siendo auditado está en cumplimiento con las guías del grupo IFPUG. La evidencia puede tomar muchas formas diferentes: el conteo de Puntos Función, la documentación del sistema, las conversaciones con desarrolladores y usuarios, las entrevistas con personas que llevaron a cabo el conteo original, etc... El auditor reúne y acumula la evidencia, y saca conclusiones.

Por supuesto el conteo de Puntos Función en sí puede ser usado como evidencia, pero usarlo sólo sería inadecuado. Es imposible determinar la precisión de un conteo de Puntos Función, sin la evaluación de evidencias adicionales.

Pero... ¿cómo se lleva a cabo una auditoría de Puntos Función? Veamos... Si a un auditor se le encomienda la tarea de auditar una compañía con 500.000 Puntos Función en aplicaciones software, sería imposible revisar cada conteo. El auditor seleccionará sólo de 20 a 30 aplicaciones para practicar la auditoría. El tamaño de esta muestra variará según el auditor y según la auditoría de que se trate. La decisión de cuántos elementos tomar debe ser hecha por el auditor para cada procedimiento de auditoría. Hay varios factores que determinan el tamaño apropiado de la muestra. Los dos más importantes son las expectativas de los auditores en cuanto al número de errores, y la efectividad de los procedimientos de conteo de Puntos Función. El procedimiento que se sugieren en el *Apartado 3.6.3* pueden ayudar a determinar ambos criterios.

Adicionalmente, la evidencia debe incumbir o ser relevante en la auditoría, y el auditor debe tener la suficiente habilidad como para detectar el mayor número posible de errores en el conteo. Por ejemplo, el auditor puede determinar, durante las entrevistas, que hubo cierta confusión acerca de las Entradas Externas y de los Ficheros Externos de Interfaz. En este caso, el auditor deberá revisar la documentación disponible del sistema actual, así como el conteo de Puntos Función, para asegurarse que todas las Entradas Externas y Ficheros Externos de Interfaz fueron tratados correctamente. Otro ejemplo sería que el experto en conteo de Puntos Función nunca hubiera realizado una cuenta sobre el interfaz gráfico de una aplicación. El auditor revisaría entonces una serie de pantallas para determinar si el experto contó correctamente tales elementos (botones, menús, cuadros de diálogo, casillas de verificación, etc...).

La evidencia se debe considerar creíble y digna de confianza. Si la evidencia se considera digna de confianza, representa una gran ayuda para el auditor en la auditoría de Puntos Función. Por otra parte, si la evidencia es cuestionable, tal como una documentación incompleta o anticuada, entonces el auditor tendría que escrutar esas áreas de conteo más concienzudamente. Adicionalmente, el auditor deberá dar cuenta en el informe final de cualquier evidencia que pidió y el cliente no se la pudo proporcionar.

Toda evidencia debe ser evaluada en base a ***valuación***, ***grado de terminación***, ***clasificación***, ***rango***, ***precisión mecánica*** y ***análisis analítico***.

- **Valuación (Valuation):** Se refiere a si algunos elementos incluidos en el conteo de los Puntos Función debieron haber sido incluidos o no. Quizás el conteo original de los Puntos Función incluyó transacciones o archivos adicionales que no debieron ser incluidos.

- **Grado de Terminación (Completeness):** Se trata de incluir todas las transacciones y archivos en el conteo final de Puntos Función. Es importante que el equipo de desarrollo revise el conteo final de los Puntos de Funcionalidad para asegurarse de que todas las transacciones y archivos han sido

incluidos. Los objetivos de *valuación* y *grado de terminación* enfatizan intereses de auditoría contrarios. La *valuación* trata con sobrestimaciones potenciales y el *grado de terminación* trata con transacciones y archivos no registrados.

- **Rango (Rating):** Trata de determinar si las transacciones y los archivos fueron clasificados correctamente con grados de complejidad *Bajos*, *Medios* o *Altos*. Para completar este objetivo se debe realizar un examen detallado de los datos y archivos referenciados.

- **Precisión Mecánica (Mechanical Accuracy):** El probar la precisión mecánica involucra la revisión de una muestra de los cálculos y transferencias de información de un documento a otro. La revisión de los cálculos consiste en probar la precisión aritmética del conteo original de Puntos Función. Esto es más importante si no se usó una herramienta automática para contar los Puntos Función.

- **Análisis Analítico (Analyticial Analysis):** Este procedimiento es otra manera de validar el conteo de Puntos Función. Por ejemplo, la proporción de Entradas Externas, Salidas Externas, Consultas Externas, Ficheros Lógicos Internos y Ficheros Externos de Interfaz, puede ser comparada con la proporción de estos elementos en otras aplicaciones de características similares. También, las características generales del sistema (GSC's) pueden ser revisadas y comparadas con aplicaciones similares. Los procedimientos analíticos deben ser realizados durante las primeras fases de una auditoría, de tal manera que puedan ayudar al auditor a determinar las áreas que necesitan ser investigadas más profundamente.

Antes de realizar una auditoría o una validación de un conteo de Puntos Función, se debe seguir un procedimiento para evaluar dicho conteo. Un procedimiento, como mínimo, que debe cubrir todas las áreas mencionadas anteriormente. El procedimiento no tiene que ser un documento rígido, sino una guía para realizar la auditoría. Dicho procedimiento podría ser perfectamente el que se muestra en el *Apartado 3.6.3*.

3.6.2. Tipos de Informes:

La etapa final del proceso de auditoría es el informe de la auditoría, es decir, la comunicación de los resultados a los interesados. Los informes difieren en cuanto a su naturaleza, pero en todos los casos deberán informar a los interesados del cumplimiento de las reglas y criterios de conteo del grupo IFPUG. El auditor puede confeccionar tres tipos de informes distintos:

1. Como en todas las auditorias, el informe final más común es el ***Informe Completo Estándar de Auditoría***. Es usado en el 90% de las auditorías, y no debe ser diferente para una auditoría de Puntos Función. Este informe es usado cuando se cumplen las siguientes condiciones:

    a) Toda la documentación de los usuarios y sistemas ha sido incluida en el conteo original de Puntos Función.
    b) Se cumplieron las guías prácticas de conteo del grupo IFPUG.
    c) El conteo de Puntos Función se documentó sin problemas.

2. Otro tipo de informe de auditoría refleja las ***"condiciones que requieren una desviación".*** Hay dos condiciones que requieren una desviación del *Informe Completo Estándar de Auditoría*:

    a) Cuando el alcance del auditor ha sido restringido. Este es el caso en el que el auditor no ha reunido suficiente evidencia como para poder determinar si el conteo de Puntos Función se realizó de acuerdo a las guías de conteo de la IFPUG.

b) Cuando el conteo de Puntos Función no se realizó de acuerdo a las reglas de conteo de la IFPUG. En este caso, se debe incluir en el informe final un análisis detallado de las áreas específicas en las que no se siguieron las reglas.

3. Adicionalmente, el auditor puede emitir una ***opinión adversa*.** Una opinión adversa se usa solo cuando el auditor cree que el conteo de los Puntos Función es tan engañoso que no representa propiamente el tamaño funcional de la aplicación que se está midiendo. El auditor debe ser muy concreto en el porqué de esa conclusión.

3.6.3. Preguntas clave en la auditoría del conteo de Puntos Función:

A continuación se muestran 20 preguntas clave en la realización de una auditoría de Puntos Función:

1. ¿Fue la tarea de conteo de Puntos Función incluida en el plan general del proyecto?

Todas la actividades que el equipo de trabajo emprenda deben estar incluidas en el plan del proyecto. Hay que asegurarse de que se dedica la cantidad de tiempo necesaria para completar la tarea en su totalidad.

2. La persona encargada del conteo... ¿ha recibido entrenamiento formal en conteo de Puntos Función?

Muy a menudo, los conteos de Puntos Función son realizados por personal que no ha sido formalmente entrenado. Es deseable que la persona que realiza el conteo haya pasado el examen de certificación IFPUG. Pasar el examen no asegura que el conteo sea preciso, pero garantiza al menos un nivel mínimo de competencia.

3. ¿Se tomaron en cuenta las reglas de conteo del grupo IFPUG 4.0?

4. El experto en conteo de Puntos Función... ¿usó documentación reciente del proyecto para contar los Puntos de Funcionalidad? Si no fue así, ¿cómo de antigua era la documentación?

5. ¿El equipo de desarrollo participó en el conteo de Puntos de Funcionalidad?

El equipo de desarrollo debe estar formado por las personas más familiarizadas con respecto a la funcionalidad que ha de entregarse al usuario. Esta será la mejor fuente de información con respecto al proyecto. Frecuentemente el equipo de trabajo no tiene participación alguna cuando se realiza un conteo de Puntos Función; el experto en conteo de Puntos Función reúne alguna documentación y se sienta en su mesa de trabajo durante varios días hasta obtener un conteo de Puntos de Funcionalidad.

6. ¿Se siguieron las guías internas de conteo de Puntos Función?

7. ¿Se midió la aplicación desde el punto de vista del usuario?

8. ¿Se midió el sistema desde un punto de vista lógico, no físico?

9. El contexto establecido para el conteo de Puntos de Funcionalidad... ¿era acorde con el contexto de otras métricas? Si no es así, ¿por qué?

10. Si el conteo de Puntos Función era para tareas de mantenimiento... ¿era el contexto el mismo que el contexto de la aplicación? Si no es así, ¿por qué?

11. ¿Ha cambiado el contexto?, Si es así, ¿Por qué?

12. ¿Se usó alguna herramienta para el conteo de Puntos Función, o el conteo se realizó manualmente?

Si el conteo se realizó manualmente, se deberá revisar la aritmética usada.

13. Los porcentajes de los componentes individuales de los Puntos Función (EI, EO, EQ, ILF, EIF)... ¿son acordes con el promedio de la industria?. Si no es así, ¿hay alguna razón válida?

Si se auditan varias aplicaciones, ¿son similares los porcentajes de transacciones y archivos?

14. El inventario de transacciones (EI, EO,EQ) y archivos (ILF, EIF)... ¿ha sido revisado por el equipo de trabajo?

El error más grande cuando se cuentan Puntos Función es el error de omisión (no incluir todo lo necesario). Es importante que el equipo de desarrollo revise el conteo de Puntos Función para verificar el grado de terminación y de precisión.

15. ¿El Factor de Ajuste total es compatible con otros proyectos?

El Factor de Ajuste total (VAF) debe caer dentro del ±5 por ciento del promedio del factor de ajuste para todas las aplicaciones revisadas. Si cae fuera de este rango, se deberá incluir una explicación por escrito junto con el conteo de Puntos Función. Por ejemplo, si el VAF promedio fue de 1.05, entonces el VAF debió haber estado entre 1.0 y 1.1.

16. ¿Cada una de las 14 Características Generales del Sistema (GSC's) cae dentro de los rangos observados en otros proyectos? ¿Cada una de las Características Generales del Sistema está en torno a un punto del promedio de GSC's?.

Por ejemplo, si una GSC particular se mide como 2.0, entonces la GSC debe ser 1, 2 ó 3. Si la GSC está fuera de este rango, se deberá incluir una explicación por escrito junto con el conteo de Puntos Función.

17. ¿Se han documentado todas las suposiciones hechas en el conteo de Puntos Función?

Todas las suposiciones deben ser documentadas para que puedan ser revisadas posteriormente, en caso de que sea necesario.

18. ¿Son estas suposiciones consistentes con otros proyectos?

19. ¿Se han enviado todas las suposiciones que afectan al conteo de Puntos Función a un grupo central de Puntos de Funcionalidad?

Todas las suposiciones deberán ser revisadas por un grupo central de Puntos de Funcionalidad.

20. ¿Ha sido revisado el conteo por un especialista certificado en conteo de Puntos Función?

3.7. Los Puntos Función no son la métrica perfecta:

Cierto es que el uso de los Puntos Función está cada vez más difundido entre las organizaciones de desarrollo de software. Sin embargo, no se trata de la métrica perfecta; existen ciertas controversias y desacuerdos entre la comunidad de usuarios y, lo que es más importante, hay algo de subjetividad en el proceso de medición, sobre todo a la hora de aplicar los *factores de peso* según la complejidad del elemento medido. Dicha subjetividad existe y, por lo tanto, puede provocar desviaciones de hasta un ±10% en los resultados finales de la medición, si los cálculos son realizados por diferentes personas sobre una misma aplicación software. Podemos decir que este margen de error no es tan elevado, si tenemos en cuenta que otras técnicas de medición (como el conteo de líneas de código) pueden arrojar variaciones mucho mayores, en algunos casos de hasta el 100%.

Como en todo, los Puntos Función tienen detractores y defensores. Algunos detractores mantienen que, como los Puntos Función fueron creados para medir sistemas de información orientados a la gestión, no son adecuados para otros tipos de sistemas, como los sistemas en tiempo real, los sistemas embebidos, etc... De ahí que hayan ido apareciendo numerosas variantes de los Puntos Función como, por ejemplo, los **Puntos Característica** (Feature Points) orientados a la medición del tamaño funcional de aplicaciones científicas y de ingeniería, los **Puntos Función 3D**, los **Puntos Función de Ingeniería**, los **Puntos Objeto**, los **Puntos Función Completos** (Full Function Points), etc..., algunos de los cuales se estudian en este trabajo.

A pesar de las deficiencias que puedan presentar, los Puntos Función constituyen hoy en día la mejor forma de medir software. Utilizan un proceso de medida normalizado, lo cual permite establecer estimaciones objetivas de calidad, costes y tiempos.

4. Los Puntos Característica (Feature Points).

4.1. Introducción:

Como sabemos, los Puntos Función fueron originariamente desarrollados para medir el tamaño funcional de sistemas software orientados a la gestión. En este campo de aplicación, los Puntos Función han demostrado ser, quizás, la métrica más eficaz, lo cual ha hecho que sea una de las más utilizada en gran parte del mundo.

Pero si alteramos el campo de aplicación, es decir, si aplicamos esta métrica no a sistemas de gestión sino a otros tipos de sistemas (software para control de procesos, sistemas en tiempo real, sistemas embebidos, etc...), los Puntos Función dejan de ser la métrica perfecta: ya no se comporta de forma tan eficaz y tan óptima como con los sistemas de gestión. Surge, entonces, la necesidad de buscar una medida alternativa del tamaño funcional para estos otros tipos de aplicaciones.

Con esta intención, Caper Jones [JON86] desarrolló en 1986 un método experimental para adaptar la métrica de Puntos Función a sistemas software científicos y de ingeniería. Para evitar confusiones con los Puntos Función de Albrecht, a este nuevo método de medida del tamaño funcional se le dió el nombre de Puntos Característica[5] (Feature Points). Hasta el día de hoy, son muchas las pruebas que se han realizado con este nuevo método de medida. Los Puntos Característica se han venido utilizando con gran éxito en la medición de sistemas software muy variados: sistemas en tiempo real, sistemas embebidos, aplicaciones CAD, software para inteligencia artificial, etc... A diferencia de los sistemas de gestión, este otro tipo de sistemas tiene dos características básicas:

1. La complejidad algorítmica que implementan.
2. El escaso número de entradas y salidas que suelen tener.

Puesto que el conteo de Puntos Función no contempla para nada la complejidad algorítmica de los sistemas, esto hace que los resultados obtenidos no sean realmente significativos. En general, y para estos sistemas, el resultado obtenido tras un Análisis de Puntos Función suele ser bastante inferior (entre un 20 y un 35%) al que se obtiene con otras técnicas que sí contemplan la complejidad algorítmica.

Por ejemplo: para aplicaciones donde el número de algoritmos es mucho mayor que el número de ficheros lógicos (sistemas de control, embebidos, de tiempo real, etc...), los Puntos Característica devuelven un valor mucho más alto que los Puntos Función. Y al contrario; para aplicaciones con un gran número de ficheros lógicos y poca complejidad algorítmica (sistemas de gestión), los Puntos Característica devolverán un valor inferior al devuelto por los Puntos Función, aunque en algunas ocasiones dichos resultados pueden asemejarse bastante.

4.2. Proceso de conteo de Puntos Característica:

La métrica de Puntos Característica es un *superconjunto* de la métrica de Puntos Función. Introduce un nuevo componente, los **algoritmos**, que se añaden a los cinco ya existentes: Entradas Externas, Salidas Externas, Consultas Externas, Ficheros Lógicos Internos y Ficheros Externos de Interfaz. Un algoritmo se define como *"un problema de complejidad computacional limitada, que se incluye dentro de un determinado programa de computadora"* [PRES97]. Caper Jones propone una definición alternativa: *"Es un conjunto de reglas perfectamente definidas, que ayudan a la resolución de un determinado problema computacional"* [JON86]. Pero el tema que nos ocupa es el del conteo de Puntos Característica; en dicha métrica, el término algoritmo se definirá como *"un problema computacional de alcance y límites bien definidos, que se implementa dentro de una determinada aplicación informática"*.

[5] Ambos conceptos, *Puntos Función* y *Puntos Característica*, significan lo mismo: *funcionalidad* o *utilidad* que implementa la aplicación software.

Por defecto, a este nuevo componente (los algoritmos) se le asigna un factor de peso de 3 Puntos Característica. Además, en el conteo de Puntos Característica se reducen los pesos asignados a los Ficheros Lógicos Internos de complejidad *Media*, pasando éstos de tener un peso de 10 Puntos Función a tener un peso de 7 Puntos Característica. Por lo que respecta al resto de los componentes del sistema, se usa un único factor de peso para cada uno de ellos, como puede verse en la tabla 11.

Componente	Complejidad del componente (factor de peso)	Total
Entradas Externas	____ x 4 = ____	____
Salidas Externas	____ x 5 = ____	____
Consultas Externas	____ x 4 = ____	____
Ficheros Lógicos Internos	____ x 7 = ____	____
Ficheros Externos de Interfaz	____ x 7 = ____	____
Algoritmos	____ x 3 = ____	
Nº Total de Puntos Característica sin Ajustar (PCsA):		____
Factor de Ajuste (VAF):		x ___
Nº Total de Puntos Característica Ajustados (PCA):		_____

Tabla 11. Conteo de Puntos Característica.

Como puede verse en la tabla anterior, el método de conteo de Puntos Característica es similar al de conteo de Puntos Función (consultar éste en el Paso 7 del Apartado 3.3). En el caso que nos ocupa, para obtener el ***Número Total de Puntos Característica Ajustados*** (*PCA*) (equivalente al tamaño funcional de la aplicación), multiplicaremos el ***Factor de Ajuste*** (*VAF*) por el ***Número Total de Puntos Característica sin Ajustar*** (*PCsA*), es decir:

$$PCA = VAF * PCsA$$

4.3. Los algoritmos en el proceso de conteo de Puntos Característica.

En el apartado anterior vimos varias definiciones del concepto de 'algoritmo'. Los algoritmos varían tanto en dificultad de implementación como en complejidad computacional, por lo que han de aplicárseles factores de peso distintos. El rango de valores que se utiliza es de 1 a 10 aunque, como ya dijimos, se suele aplicar por defecto el factor de complejidad 3. Cuando los algoritmos se basan únicamente en operaciones aritméticas básicas, de poca complejidad, se asigna el valor mínimo: 1. Sin embargo, cuando se basan en complicadas ecuaciones matemáticas, operaciones con matrices, etc... se aplica el valor máximo: 10. De todo esto es fácil deducir que según aumente la complejidad algorítmica de una aplicación, así aumentará el valor final en Puntos Característica, aunque dicha correlación no es totalmente lineal.

De nuevo aparece una cierta subjetividad a la hora de estimar cómo de complejo es un algoritmo. Para tratar de evitarla en lo posible, se atiende a los dos aspectos siguientes:

1. El número de pasos de cálculo o reglas de que consta el algoritmo.
2. El número de factores o datos que necesita el algoritmo para trabajar.

Estos criterios no son ni mucho menos determinantes ni definitivos, puesto que la técnica de conteo de Puntos Característica está aún en fase experimental.

Por otra parte, para determinar qué algoritmos son realmente significativos en el proceso de conteo y, por tanto, cuáles es preciso incluir en la cuenta, se han definido una serie de reglas o condiciones que dichos algoritmos deberán cumplir. Se trata de las siguientes:

1. Los algoritmos deberán estar pensados para problemas resolubles y con alcance y límites bien definidos.
2. Los algoritmos deberán ser finitos en tiempo de ejecución, es decir, su ejecución deberá concluir en tiempo finito.
3. Deberán ser precisos y no ambiguos.
4. Deberán admitir una entrada o valores de comienzo, y devolver una salida o valores resultado.
5. Un algoritmo podrá incluir uno o varios subalgoritmos, o bien llamar a otros algoritmos.
6. Todos los pasos de que conste deberán poderse implementar en la máquina, haciendo uso de las sentencias típicas de la programación estructurada (if-then-else, do-while, case, etc...).

Una vez más hemos de recordar que continuamente se tratan de ampliar, mejorar y definir más explícitamente estos criterios, ya que el conteo de Puntos Característica no es, hoy por hoy, una técnica totalmente madura.

4.4. Puntos Función vs Puntos Característica.

Es muy común que surja la siguiente pregunta: ¿Cuándo utilizar los Puntos Función y cuándo los Puntos Característica?

La respuesta es clara, aunque no siempre fácil de detectar: si la aplicación que pretendemos medir tiene un alto componente algorítmico (un gran número de algoritmos) o éstos, aun siendo pocos, implican una elevada complejidad computacional, se optará por el uso de Puntos Característica como el método más apropiado para medir su tamaño funcional. Por el contrario, si en la aplicación a medir aparecen pocos algoritmos (o de escasa importancia) y un alto número de entradas, salidas, consultas y ficheros lógicos, se optará por el conteo de Puntos Función como la opción más apropiada.

A día de hoy, son pocos los sistemas cuyo tamaño funcional se ha medido con el conteo de Puntos Característica. Debido a eso, los resultados disponibles son realmente escasos, pero suficientes como para sacar ciertas conclusiones. Obsérvese la tabla 12, donde se muestran una serie de ratios comparativos entre aplicaciones medidas en Puntos Función y en Puntos Característica:

Tipos de sistemas	Ratios en...	
	Puntos Función	Puntos Característica
Orientados a la gestión, con proceso por lotes.	1.00	0.80
Orientados a la gestión, interactivos.	1.00	1.00
Bases de Datos on-line.	1.00	1.00
Para el control de procesos.	1.00	1.28
Embebidos / En tiempo real.	1.00	1.35
Para automatización industrial.	1.00	1.50

Tabla 12. Ratios comparativos entre Puntos Función y Puntos Característica, para distintos sistemas.

Como podemos ver, para aplicaciones de gestión con *proceso por lotes*, ofrece mejores resultados la estimación con Puntos Función que la estimación con Puntos Característica. Para aplicaciones de gestión interactivas y aplicaciones de bases de datos on-line, ambos indicadores producen los mismos resultados, es decir, es indiferente la utilización de uno u otro. Para el resto de aplicaciones (control de procesos, sistemas embebidos en tiempo real, etc...), la estimación con Puntos Característica resulta ser la opción más apropiada.

5. La Métrica *Bang*.

5.1. Introducción:

Como sabemos, es muy habitual que en las primeras etapas de los procesos de desarrollo software comiencen a realizarse predicciones y estimaciones acerca del coste y la duración de dichos procesos. Pero este tipo de estimaciones requiere (más aún en las primeras etapas) que hayan sido totalmente identificados los requisitos de usuario y que las especificaciones de dichos requisitos hayan quedado perfectamente definidas. En definitiva, debe existir el conjunto de especificaciones formales del sistema, el cual nos ayudará también a entender el alcance, la envergadura y la complejidad del sistema software al que nos enfrentamos.

Para conseguir esto, y recurriendo al *diseño estructurado*, se suele utilizar un *modelo compuesto* del sistema, obtenido de la unión de otros tres modelos: el funcional (qué hace el sistema), el de datos (qué datos utiliza) y el de comportamiento (cómo se comporta el sistema). Esta solución nos permitirá obtener una representación gráfica, concisa y fácil de entender de cuáles son las especificaciones del sistema. Se trata del *modelo de requisitos del software* o *modelo de especificación,* altamente estructurado, y representativo de la funcionalidad del sistema. Las *métricas de especificación,* entre las que se encuentra la métrica *Bang,* serán las encargadas de captar y medir dicha funcionalidad.

El *modelo de especificación* expresa el conjunto global de requisitos de la aplicación software, pero no hace alusión a ninguna forma particular de abordarlos. Por ello, un análisis cuantitativo del modelo proporcionará una medida de la funcionalidad que deberá implementar el software, tal y como es entendida por el usuario (es decir, desde el punto de vista del usuario/cliente de la aplicación), y sin tener en cuenta ningún requisito técnico de implementación o calidad.

En este marco de aplicación es donde encaja la métrica *Bang.* Propuesta inicialmente por Tom DeMarco en el año 1982 [MARCO82], la métrica *Bang* es una métrica orientada a la función, indicativa del tamaño funcional de un sistema, y totalmente independiente de todo tipo de cuestiones de implementación. Para su utilización será necesario, en primer lugar, determinar el tamaño del *modelo de especificación*. Es lo que veremos en los siguientes apartados.

5.2. Componentes elementales del modelo de especificación:

Un componente del modelo de especificación se considera *elemental* (o *primitivo*) si no puede dividirse en otros más pequeños. Cada parte del modelo de especificación (modelo funcional, modelo de datos y modelo de comportamiento) se divide y descompone reiteradamente hasta llegar al nivel más bajo de descomposición, donde todos los componentes son elementales. Dependiendo de qué parte del modelo es la que se está dividiendo, obtendremos unos componentes elementales u otros. DeMarco contabiliza, en total, seis tipos distintos de componentes, según se muestra a continuación:

1. **Primitivas funcionales:** Son los componentes elementales que se derivan de sucesivas descomposiciones del modelo funcional (descomposiciones de los procesos de un *diagrama de flujo de datos*, por ejemplo). Ejemplo de primitiva funcional es cualquier proceso indivisible que transforma un dato de entrada en un dato de salida. Obviamente, las primitivas funcionales forman parte del modelo funcional.
2. **Datos elementales:** Son los datos concretos e indivisibles que forman parte del *diccionario de datos* del modelo funcional. Datos elementales son: números, variables, cadenas, etc...

3. **Objetos:** Son los componentes elementales que se derivan de sucesivas descomposiciones del modelo de datos (descomposiciones de un *diagrama E/R*, por ejemplo). Un objeto es, conceptualmente, un conjunto de datos caracterizados por los mismos atributos, y que constituyen una *entidad*.
4. **Interrelaciones:** Son los componentes elementales derivados de las relaciones (conexiones) existentes entre unos objetos y otros. Forman parte del modelo de datos del sistema.
5. **Estados:** Son los componentes elementales que se derivan de sucesivas descomposiciones del modelo de comportamiento (descomposiciones de un *diagrama de estados/transiciones*, por ejemplo). Lógicamente, los estados forman parte del modelo de comportamiento.
6. **Transiciones:** Son las acciones que hacen pasar al sistema de un estado a otro del modelo de comportamiento.

Como vemos, se obtienen dos componentes elementales por cada uno de los modelos del sistema. Pues bien... contando el número de componentes elementales que aparece en el modelo de especificación del sistema, obtenemos los datos de partida para utilizar la métrica *Bang*. Estaríamos hablando, entonces, de seis cuentas sencillas. Sin embargo, DeMarco añade otras seis más. En total son:

Componente elemental:	Descripción:
FP	Número de **primitivas funcionales** del modelo funcional del sistema.
FPM	Número de **primitivas funcionales modificadas** (primitivas funcionales externas al sistema que deben modificarse para adaptarlas al nuevo sistema).
DE	Número de **datos elementales** del sistema.
DEI	Número de **datos elementales de entrada** al sistema.
DEO	Número de **datos elementales de salida** del sistema.
DER	Número de **datos elementales *retenidos*** por el sistema (datos almacenados dentro del sistema).
OB	Número de **objetos** del modelo de datos del sistema.
RE	Número de **interrelaciones** del modelo de datos del sistema.
ST	Número de **estados** del modelo de comportamiento del sistema.
TR	Número de **transiciones** del modelo de comportamiento del sistema.
TC_i	Número de ***tokens*** implicados en la *i-ésima* primitiva funcional (tanto de entrada como de salida). Un *token* es un dato o conjunto de datos que la primitiva funcional trata como un todo, es decir, no requiere ser dividido para que la primitiva funcional pueda utilizarlo. El parámetro TC_i debe calcularse para cada primitiva funcional.
RE_i	Número de interrlaciones en las que está involucrado el *i-ésimo* objeto del modelo de datos. Este parámetro debe calcularse para cada objeto del modelo de datos.

***Tabla 13**. Componentes elementales que han de contabilizarse para aplicar la métrica Bang.*

Ni que decir tiene que a la hora de contabilizar el número de componentes elementales del modelo de especificación, deberemos comprobar exhaustivamente que dicho modelo no tenga ningún tipo de redundancias entre sus componentes. Si hubiera redundancias, existiría la posibilidad de contar dos o más veces el mismo elemento, con lo que se desvirtuaría la medición.

5.3. Clasificación de los sistemas:

Una vez contabilizados todos los componentes elementales del modelo de especificación, el siguiente paso será elegir uno de ellos como indicador principal (sobre el que se basará el cálculo del tamaño funcional) y utilizar los otros como modificadores. Para la mayoría de los sistemas, este indicador principal suele ser el componente FP, es decir, las primitivas funcionales del sistemas. Evidentemente, no todas las primitivas funcionales son iguales; unas suelen ser más difíciles de implementar que otras, por lo que requieren un mayor esfuerzo de desarrollo. Pero consiguiendo corregir y equiparar estas diferencias, el valor contabilizado para FP será un indicador bastante preciso de la medida del tamaño funcional de la aplicación.

Ahora bien, el factor FP será lo que hemos llamado *indicador principal*, pero no en todos los casos. Lógicamente, lo será en aquéllos casos en los que el sistema en estudio tenga un alto componente funcional, es decir, su modelo funcional (DFD's) sea mucho más importante y mucho más extenso que su modelo de datos (esquema E/R) y que su modelo de comportamiento (diagrama de transición de estados).

Está claro, entonces, que el factor FP no podrá ser el *indicador principal* en aquéllos casos en los que, por ejemplo, el sistema esté fuertemente orientado a los datos. Es decir, que puede expresarse completamente en términos de *'datos con los que trabaja'* e *'interrelaciones entre esos datos'*, más que en términos de *'operaciones que hace con los datos'*. En estos casos, el *indicador principal* apropiado será el factor OB.

Pero.... ¿cuándo un sistema está fuertemente orientado a los datos, y cuándo lo está al proceso y transformación de esos datos?. DeMarco [MARCO82] propone dos criterios para clasificar unos sistemas y otros. Son los siguientes:

- Si **RE / FP < 0.7** entonces el sistema está fuertemente orientado a proceso y transformación de datos.
- Si **RE / FP > 1.5** entonces el sistema está fuertemente orientado a los datos.
- Si **(RE / FP ≥ 0.7)** y **(RE / FP ≤ 1.5)** entonces el sistema es *híbrido*, es decir, se trata de un sistema con un importante componente funcional y de datos.

...siendo **RE** el número de interrelaciones del modelo de datos del sistema, y **FP** el número de primitivas funcionales del modelo funcional del sistema.

- Ratio **DEO / FP**: Es un ratio indicativo del *trasiego* de datos desde dentro del sistema hacia fuera. Los sistemas comerciales o de gestión suelen tener un ratio DEO / FP muy elevado, mientras que los sistemas científicos o de ingeniería (más orientados al cálculo) suelen tener un ratio DEO / FP pequeño. No existe un valor concreto del ratio DEO / FP, que nos permita catalogar unos sistemas en un dominio o en otro.

El factor DEO es el número de datos elementales de salida del sistema.

La siguiente figura muestra cómo pueden catalogarse los sistemas software, en función de estos dos ratios:

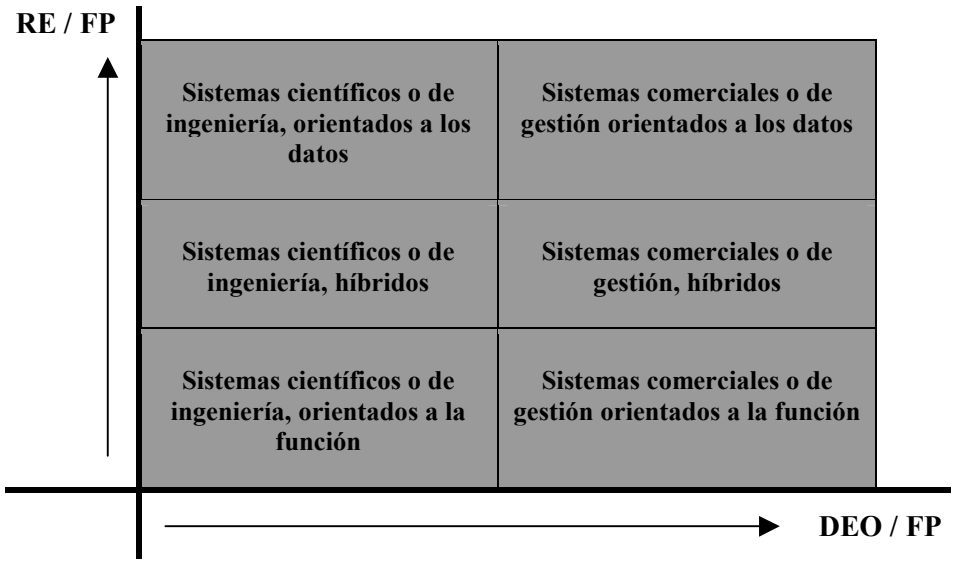

Figura 5. Catalogación de los sistemas software, en función de los ratios DEO / FP y RE / FP.

5.4. Formulación de la métrica Bang:

Como hemos visto en el apartado anterior, DeMarco distingue tres tipos de sistemas software: los orientados a los datos, los orientados a la función (proceso y transformación de esos datos) y los híbridos. Por lo tanto, la métrica Bang se formula de tres formas distintas, en función del sistema en estudio:

A. Para sistemas orientados a la función:

En los sistemas orientados a la función, el componente elemental más importante son las primitivas funcionales. Pero, como hemos dicho antes, unas primitivas funcionales pueden ser más grandes o más complejas de desarrollar que otras, por lo que hemos de corregir de alguna forma estas variaciones.

Durante la construcción del modelo de especificación (y más concretamente, del modelo funcional), nos encontramos siempre con el problema de cuando dividir un proceso en otros subprocesos, o cuando dejarlo como proceso elemental. Este hecho hará distintas personas tomen soluciones distintas por lo que, al final, la contabilización de primitivas funcionales (factor FP), que es lo que nos interesa, variará en unos casos y en otros. Necesitamos, entonces, alguna regla a seguir:

"En el proceso de descomposición de un modelo, dejaremos un componente como primitivo o elemental, siempre que no sea posible su división o descomposición, o siempre que su división o descomposición no reduzca el factor TC_{medio}".

El factor TC_{medio} a que se refiere viene dado por:

$$TC_{medio} = \frac{\sum TC_i}{FP}$$

...y es el número medio de *tokens* por primitiva funcional. (Un *token* es un dato o conjunto de datos que la primitiva funcional trata como un todo, es decir, no requiere ser dividido para que la primitiva funcional pueda utilizarlo).

Pues bien... hemos establecido así una regla que nos permite realizar una partición uniforme del modelo funcional. Pero esto no evita que unas primitivas funcionales sean más grandes que otras, en cuanto a tamaño. Se necesita entonces un método de corrección que equipare el tamaño de unas con el de otras. Para ello, deberemos entender una primitiva funcional como un proceso elemental en el que se transforman unos datos de entrada en otros de salida. El factor a utilizar será el TC_i, es decir, el número de *tokens* involucrados en la *i-ésima* primitiva funcional, bien sean de entrada a la primitiva, bien sean de salida.

Conociendo entonces el valor de TC_i para cada primitiva funcional, podemos aplicar la siguiente regla (propuesta por Halstead en 1977 [HALS77]), para conocer cuál sería el tamaño corregido de cada primitiva funcional ($CFPI_i$):

El tamaño de la primitiva 'i' es proporcional a $TC_i * \log_2 (TC_i)$

Por lo tanto, la fórmula propuesta es: $CFPI_i = TC_i * \log_2 (TC_i)$

Y una vez que conocemos el tamaño corregido de cada primitiva funcional, lo único que queda es sumar dichos valores para saber cuál es el **tamaño funcional total** de nuestro sistema. Al valor obtenido se le llama CFP (Corrected Functional Primitive) y representa el tamaño total corregido de todas las primitivas funcionales del sistema:

$$CFP = \sum_i CFPI_i$$

Nos queda, por último, ver qué es lo que sucede en el caso de que haya mucha variación en cuanto a la complejidad de las primitivas funcionales de nuestro sistema, es decir, que unas sean más complejas y más costosas de implementar que otras, lo cual es muy habitual. En estos casos, DeMarco propone clasificar las primitivas funcionales en diferentes categorías y asignarle a cada una de ellas un *peso* o factor de corrección en función de la complejidad que presenten. Según DeMarco, existirán primitivas funcionales de los tipos que se indican en la primera columna de la tabla 14, y sus pesos asociados serán los indicados en la segunda columna. Hay que señalar que estos pesos son dependientes del entorno de desarrollo y están gravemente afectados por el lenguaje de programación que se utilice para la implementación. Además, DeMarco indica dichas categorías y pesos son orientativos, y que sería conveniente que cada experto desarrollara los suyos propios.

Categoría	Peso
Separación	0.6
Unión	0.6
Encauzado de datos	0.3
Actualización simple	0.5
Almacenamiento	1.0
Edición	0.8
Verificación	1.0
Manipulación de texto	1.0

Categoría	Peso
Sincronización	1.5
Generación de salidas	1.0
Muestreo	1.8
Análisis tabular	1.0
Aritmética	0.7
Iniciación	1.0
Cálculo matemático	2.0
Manipulación de dispositivos	2.5

Tabla 14. Categorías y factores de corrección, según la complejidad de las primitivas funcionales.

B. Para sistemas orientados a los datos:

Cualquier sistema orientado a los datos será un sistema que, con toda seguridad, implementará o gestionará una base de datos. Por lo tanto, la mayoría del esfuerzo de desarrollo para estos sistemas estará destinado a la implementación de la propia base de datos. Parece obvio, entonces, que el *indicador principal* que se utilice para la medida del tamaño funcional sea el factor OB, es decir, el número de **objetos** del modelo de datos del sistema (habitualmente un modelo E/R, en el que *'objeto'* equivaldría a *'entidad'*).

Pero, al igual que sucedía antes, es necesario un factor de corrección que contemple el hecho de que unos *objetos* son más costosos de implementar que otros. DeMarco propone los siguientes factores de corrección o pesos, en función de la interrelación de esos objetos con otros (factor RE_i, o número de **interrelaciones** del modelo de datos del sistema en las que está involucrado el *objeto i*). A estos factores de corrección se les llama COBI (Corrected Object Increment), y deberán ser asignados a todos los objetos (entidades) del modelo de datos del sistema:

RE_i	COBI
1	1.0
2	2.3
3	4.0
4	5.8
5	7.8
6	9.8

Tabla 15. Factores de corrección (pesos) para los objetos de un sistema, en función de sus interrelaciones con otros.

Al igual que antes, los valores COBI no son realmente precisos, puesto que dependen del entorno de desarrollo y de las herramientas que se utilicen para implementar la base de datos. Debido a ello, sería conveniente determinar nuevos valores, según las necesidades del momento.

El valor resultante para la métrica Bang, en el caso que nos ocupa, será entonces la suma de los valores COBI de cada objeto (entidad) del sistema, es decir:

$$COB = \sum_i COBI_i$$

...representando *COB* (Corrected Object) el **tamaño funcional total** de la aplicación medida.

C. Para sistemas híbridos:

En el caso de sistemas híbridos, DeMarco propone calcular independientemente las dos métricas Bang comentadas en los apartados A y B anteriores. Es decir, se trata de:

- Considerar el sistema híbrido como un sistema orientado a la función, para calcular así la primera métrica Bang, y...
- Considerar el sistema híbrido como un sistema orientado a los datos, para calcular así la segunda métrica Bang.

Obtenemos así dos medidas del tamaño funcional, una que representa el esfuerzo de desarrollo de la parte funcional, y otra que representa el esfuerzo de desarrollo de la parte orientada a los datos.

6. El modelo de estimación COCOMO II.

6.1. Introducción:

En el año 1981, Barry Boehm [BOEHM81] presenta el *Modelo Constructivo de Coste* (COCOMO), como una jerarquía de modelos de estimación para el software. En aquella época, este modelo era perfectamente aplicable a los procesos de desarrollo software del momento. Pero en casi 20 años que han pasado desde entonces, las cosas han cambiado bastante, y se ha hecho necesaria una readaptación del modelo de tal forma que contemple las características de los procesos de desarrollo software actuales. Así las cosas, el modelo COCOMO se *'reinventó'* de nuevo a principios de los años '90, volviendo a nacer con el nombre de COCOMO II[6]. Se trata, entonces, de una revisión del modelo original, adaptado a los cambios acontecidos en el desarrollo profesional de aplicaciones software, desde los años '70 a esta parte.

Las nuevas características que introduce, su adaptación a las nuevas metodologías de desarrollo y la posibilidad de utilizarlo prácticamente en cualquiera de las etapas del ciclo de de vida del software (incluidas las primeras etapas, como modelo de estimación de la funcionalidad y el esfuerzo de desarrollo), hará que el modelo COCOMO II se convierta en uno de los más populares y utilizados a lo largo de todo el mundo.

6.2. El modelo COCOMO 81:

La jerarquía de modelos propuesta por Boehm en 1981, consta de los 3 siguientes:

- **El modelo COCOMO básico.** Se trata de un modelo univariable estático que calcula el esfuerzo y el coste del desarrollo de un proceso software, en función del tamaño del programa expresado en líneas de código (LDC) estimadas.
- **El modelo COCOMO intermedio.** Calcula el esfuerzo de desarrollo de una aplicación software, en función del tamaño del programa y de un conjunto de *"conductores de costes"*, que incluyen la evaluación subjetiva del producto, del hardware, del personal y de los atributos del proyecto.
- **El modelo COCOMO avanzado.** Incorpora todas las características del modelo intermedio y lleva a cabo una evaluación del impacto de los conductores de coste en cada fase del proceso de desarrollo software (análisis, diseño, etc...).

Además, Boehm clasifica los proyectos software en tres tipos distintos:

- De tipo **Orgánico**, que son proyectos software relativamente pequeños y sencillos, y similares a otros ya desarrollados anteriormente.
- De tipo **Semi-Acoplado**, que son proyectos software intermedios en tamaño y en complejidad.
- De tipo **Empotrado**, que son proyectos software que deberán ser desarrollados en un entorno muy restringido de hardware, software y restricciones operativas.

[6] En un primer momento, se le dio el nombre de COCOMO 2.0, aunque más tarde se optó por renombrar ambos modelos, adoptando la siguiente nomenclatura: COCOMO 81 para el modelo original, y COCOMO II para la nueva revisión del modelo.

Tras la jerarquía de modelos y la clasificación de los proyectos software, Boehm formula las ecuaciones apropiadas para la estimación de la duración y el tamaño de las aplicaciones software, así como el conjunto de *atributos conductores de coste*, necesarios para la estimación de estos parámetros con el modelo COCOMO intermedio. No reproducimos aquí ni las ecuaciones ni los *conductores de coste*, ya que nuestro objetivo es estudiar el modelo COCOMO II. Sirva esto, entonces, como breve referencia a los fundamentos del modelo COCOMO 81.

6.3. El modelo COCOMO II:

Boehm, Clark, Horowitz, Westland, Madachy, and Selby [BOEHMN *et al*, 1995] reconocieron las limitaciones del modelo COCOMO 81, los beneficios de la utilización del análisis FPA y la importancia de poder realizar estimaciones precisas sobre las nuevas metodologías de desarrollo que existen hoy en día. Debido a ello apareció el nuevo modelo COCOMO II, resultado de la adaptación del modelo original de Boehm a las características y necesidades de los procesos de desarrollo software actuales.

Al igual que el modelo COCOMO 81 comentado en el punto anterior, COCOMO II está dividido en otros tres modelos distintos:

- El modelo de **Composición de la Aplicación.** Contempla las características de los proyectos software en los que se han utilizado las nuevas herramientas de desarrollo (sobre todo las orientadas a la construcción de interfaces gráficos). Se basa en lo que se ha dado en llamar ***Puntos Objeto*.**
- El modelo de **Diseño Preliminar.** Permitirá la estimación de costes y duración de proyectos antes de haber determinado completamente su estructura (es decir, es de aplicación en las primeras etapas del ciclo de desarrollo). Utiliza un nuevo conjunto de *atributos conductores de costes*, y para él se definen nuevas ecuaciones para estimación. Se basa en las métricas de **Puntos Función no Ajustados** y en las ***Kilo–líneas de código fuente*** (KSLOC).
- El modelo de **Post-Arquitectura.** Se trata del modelo más detallado de COCOMO II. Es de aplicación una vez que la arquitectura del sistema empieza a tomar forma (durante las fases de diseño). Incorpora nuevas ecuaciones y nuevos *atributos conductores de costes*.

En los apartados que siguen trataremos más en detalle los **puntos objeto**, como medida del tamaño funcional. Como hemos visto antes, es la métrica que se utiliza en el modelo de *composición de la aplicación* de COCOMO II.

6.4. Los Puntos Objeto como medida del tamaño funcional:

La estimación con Puntos Objeto es una técnica relativamente nueva, cuyo campo de aplicación es doble:

- Por una parte, es útil en aquellas aplicaciones que puede desarrollarse con un alto componente de reutilización de software, es decir, como si se tratara de tomar componentes ya desarrollados y, al estilo de un puzzle, ensamblarlos y adaptarlos hasta conformar la nueva aplicación. Es lo que se ha dado en llamar *Composición* o *Ensamblaje de Aplicaciones*.
- Por otra parte, es útil también en aplicaciones desarrolladas por herramientas RAD (Rapid Application Development), las cuales generan automáticamente (y con el mínimo esfuerzo), todo tipo de componentes software para el sistema que se está desarrollando (interfaces gráficos, prototipos operativos, etc...).

Según esto, los Puntos Objeto serán una métrica del tamaño funcional de una aplicación software, en función del número de pantallas generadas, informes producidos y, en general, cualquier módulo generado por sistemas automáticos de desarrollo (RAD) que haya sido *integrado* en la aplicación. Al igual que sucedía con los Puntos Función, los Puntos Objeto podrán utilizarse también para estimar el esfuerzo de desarrollo de una aplicación concreta, perteneciente al ámbito de la *Composición* o *Ensamblaje de Aplicaciones.*

El proceso a seguir para la estimación en Puntos Objeto tiene cierta similitud con el proceso de conteo de Puntos Función, sólo que se tiene en cuenta un factor adicional: el % de componentes reutilizados que integrarán la aplicación, es decir, qué % de los componentes serán tomados de aplicaciones previamente desarrolladas. Este será el factor de ajuste para el conteo de Puntos Objeto, de igual manera que lo era el *VAF* para el conteo de Puntos Función.

Antes de comentar los cinco pasos de que consta el proceso de conteo de Puntos Objeto, mostraremos el significado de la terminología utilizada:

- **NOP:** (New Object Points). Es el número total de Puntos Objeto Ajustados.
- **srvr:** Es el número de tablas de datos del servidor (mainframe o equivalente) utilizadas por las pantallas de interfaz o por los informes (reports) generados por la aplicación.
- **clnt:** Es el número de tablas de datos del cliente (ordenador personal o estación de trabajo) utilizadas por las pantallas de interfaz o por los informes (reports) generados por la aplicación.
- **%r:** Es el porcentaje de pantallas, informes y módulos 3GL[7] que han sido reutilizados, es decir, que provienen de otras aplicaciones previamente desarrolladas.

Los cinco pasos para el conteo de Puntos Objeto son los siguientes:

Paso 1: Contabilización de objetos.

En este primer paso de deberán contar los objetos del sistema que resultan de interés para la estimación, es decir, pantallas de interfaz, informes (reports) y módulos desarrollados en lenguajes 3GL. Por lo tanto, el concepto de *objeto* en el modelo COCOMO II no tiene porqué estar necesariamente relacionado con el concepto de *objeto* de la Programación Orientada a Objetos.

Paso 2: Clasificación de los objetos.

Los objetos identificados en el Paso 1 deberán clasificarse en cuanto a su nivel de complejidad. Existen 3 niveles posibles: complejidad *baja*, *media* y *alta* en función, principalmente, de los atributos *srvr* y *clnt* comentados anteriormente. La siguiente tabla indica cómo han de clasificarse los objetos:

Para pantallas de interfaz				**Para informes (reports)**			
Nº de vistas que contiene	**Nº y origen de las tablas de datos**			**Nº de secciones que contiene**	**Nº y origen de las tablas de datos**		
	Total <4 (<2 srvr y <3 clnt)	**Total < 8 (2-3 srvr y 3-5 clnt)**	**Total +8 (>3 srvr y >5 clnt)**		**Total <4 (<2 srvr y <3 clnt)**	**Total < 8 (2-3 srvr y 3-5 clnt)**	**Total +8 (>3 srvr y >5 clnt)**
<3	Baja	Baja	Media	**0 - 1**	Baja	Baja	Media
3 - 7	Baja	Media	Alta	**2 – 3**	Baja	Media	Alta
>8	Media	Alta	Alta	**+4**	Media	Alta	Alta

Tabla 16. Clasificación de los objetos del sistema (pantallas de interfaz e informes).

[7] Módulos desarrollados en lenguajes de 3ª generación.

Como vemos, la tabla anterior sólo clasifica a las pantallas de interfaz y a los informes (reports), Los módulos desarrollados en lenguajes 3GL se considerarán siempre de complejidad *Alta*.

Paso 3: Asignación de *pesos* a los objetos del sistema.

A cada objeto del sistema, una vez clasificado, hay que asignarle un valor concreto en Puntos Objeto (*peso*) que representará el esfuerzo requerido para su implementación. Los valores propuestos por el modelo COCOMO II son los que se muestran en la tabla siguiente:

Tipo de Objeto	Complejidad del objeto		
	Baja	Media	Alta
Pantallas de interfaz.	1	2	3
Informes (reports).	2	5	8
Componentes 3GL	-	-	10

Tabla 17. Pesos (en Puntos Objeto), en función de la complejidad de los objetos del sistema.

Paso 4: Cálculo del número total de Puntos Objeto sin Ajustar:

El número total de Puntos Objeto sin Ajustar (*POsA*) será la suma de todos los Puntos Objeto asignados a cada uno de los componentes del sistema (PO_i), es decir:

$$POsA = \sum_i PO_i$$

Paso 5: Cálculo del número total de Puntos Objeto Ajustados:

Para calcular el número total de Puntos Objeto Ajustados (*POA*), deberemos multiplicar el número total de Puntos Objeto sin Ajustar (*POsA*) por el % de componentes reutilizados que se espera incorporar al sistema (*%r*). El resultado obtendio (*POA*) también se conoce con el nombre de NOP (New Object Points), y representa el tamaño funcional de la aplicación en estudio.

$$POA \equiv NOP = POsA * \frac{(100 - \%r)}{100}$$

Una extensión del modelo COCOMO II nos permitirá, incluso, determinar ratios de productividad tomando en consideración tanto las posibilidades de la aplicación RAD utilizada, como la experiencia y capacidades del equipo de desarrollo de la aplicación. Se utiliza la siguiente ecuación:

$$Esfuerzo = \frac{NOP}{PROD}$$

...obteniéndose el esfuerzo de desarrollo en Personas-Mes. PROD representa la productividad del equipo de desarrollo y del entorno, y se obtiene la siguiente tabla:

Experiencia y capacidades del equipo de desarrollo:	**Muy baja**	**Baja**	**Normal**	**Alta**	**Muy Alta**
Posibilidades del ICASE (Integrated Computer Aided Software Enviroment)	**Muy baja**	**Baja**	**Normal**	**Alta**	**Muy Alta**
PROD:	4	7	13	25	50

Tabla 18. Ratios de productividad según equipo y entorno de desarrollo.

7. Puntos Función Completos (Full Function Points).

7.1. Introducción:

Como vimos en el apartado correspondiente, el análisis FPA era de aplicación en sistemas más bien orientados a la gestión. En estos sistemas, los Puntos Función han sido ampliamente utilizados habiéndose obtenido excelentes resultados. Pero... ¿qué ocurre si nuestro sistema no es orientado a la gestión, sino que se trata de un sistema de tiempo real, o un sistema empotrado? Numerosos autores han detectado las deficiencias del método FPA en estos ámbitos de aplicación, hasta el punto en que ha sido desplazado por otras aproximaciones a la medida del tamaño funcional.

Así las cosas, los Puntos Función Completos (en adelante FFP) aparecen en el mundo de las métricas funcionales, demostrando su eficiencia en la medición de sistemas de control, de tiempo real y embebidos. Desde su aparición en 1997, su utilización se ha difundido a lo largo de todo el mundo (Japón, Canadá, EE.UU., Reino Unido, Australia...).

7.2. El software de tiempo real:

Existen numerosas definiciones sobre lo que es un sistema en tiempo real. Veamos dos de ellas:

- Stankovic & Ramamritham (1988): *"Los sistemas en tiempo real se definen como aquellos sistemas en los que su **eficacia** depende no sólo de la bondad de los resultados por ellos obtenidos, sino también del **tiempo** que tardan en obtener esos resultados"*.
- IEEE (1990): *"Un sistema en tiempo real es aquél en el que los cómputos se realizan en el mismo momento en el que tiene lugar un proceso externo al sistema, de tal manera que los resultados de dichos cómputos puedan servir para controlar, monitorizar o actuar sobre dicho proceso externo, en un **tiempo** prudencial"*.

Como vemos, en las dos definiciones aparece un factor clave en los sistemas de tiempo real: el **tiempo**. Y en la última de ellas aparece un segundo factor: la **interacción del sistema con entidades externas.** El propósito de esta interacción es: bien obtener información de entidades externas al sistema, bien actuar o controlar dichas entidades externas, o bien las dos cosas. Y puesto que aparece el factor tiempo como uno de los más importantes, los sistemas de tiempo real deberán tratarlo adecuadamente, para lo cual estarán afectados por una serie de restricciones temporales muy explícitas. El no cumplir dichas restricciones, provocará un malfuncionamiento del sistema.

7.3. Limitaciones del análisis FPA con sistemas de tiempo real:

El FPA presenta dos limitaciones básicas a la hora de medir sistemas software de tiempo real:

1. **En cuanto a los datos:** En los sistemas de tiempo real se distinguen dos tipos principales de estructuras de datos de control: *Grupo Lógico de Ocurrencia Múltiple*, los cuales pueden tener más de una instancia del mismo tipo de registro; y *Grupo Lógico de Ocurrencia Única*, los cuales tienen una y sólo una instancia de un tipo de registro. Generalmente, los sistemas de tiempo real tienen un número muy elevado de Grupos Lógicos de Ocurrencia Única, que suelen ser difíciles de agrupar en Ficheros Lógicos Internos o en Ficheros Externos de Interfaz (que serían los equivalentes en el análisis FPA). Se requiere, por tanto, una extensión de las reglas de identificación de los Ficheros Lógicos Internos y los Ficheros Externos de Interfaz, para una correcta medición de los Grupos Lógicos de Ocurrencia Única.

2. **En cuanto a las transacciones:** Los procesos software de tiempo real tienen una característica transaccional en común: el número de subprocesos en los que se pueden dividir varía mucho de unos procesos a otros. Por lo tanto, deberá contemplarse que unos procesos tendrán sólo unos pocos subprocesos y que otros, en cambio, tendrán un gran número de ellos. Los Puntos Función tradicionales se basaban más en el número de procesos elementales del sistema que en el número de subprocesos.

7.4. Extensión del modelo FPA para sistemas de tiempo real:

Como ya sabemos, el análisis FFP es una extensión del análisis FPA, incorporando nuevas reglas y criterios para la medición de sistemas en tiempo real, entre otros. Y como extensión del FPA que es, el FFP incorporará todas las reglas de conteo del FPA (definidas por la IFPUG), más algunas otras nuevas relacionadas con el control. Este nuevo aspecto, el control, es abordado por el FFP definiendo nuevos tipos de funciones:

7.4.1. Tipos de funciones en el análisis FFP:

Para medir adecuadamente las características de los subprocesos que pueden existir en un sistema, es necesario considerar no sólo los procesos definidos en el análisis FPA (procesos elementales), sino también considerar los propios subprocesos. Y para ello, el análisis FFP introduce nuevas funciones de datos y de transacción. En total, los componentes que distingue dentro de un sistema de tiempo real, son los siguientes:

- **Grupo Lógico de Control Interno (ICLG):** Es un grupo de datos de control lógicamente relacionados, y actualizados por la aplicación que se está midiendo. Se identifican desde una perspectiva puramente funcional[8].
- **Grupo Lógico de Control Externo (ECLG):** Es un grupo de datos de control lógicamente relacionados, y usados (pero no actualizados) por la aplicación que se está midiendo. Se identifican desde una perspectiva puramente funcional[8].
- **Entrada Externa de Control (ECE):** Es un único subproceso cuyo cometido es procesar los datos de control que vienen desde fuera del sistema, evitando la actualización de datos, que será realizada por otro componente (el ICW). Se corresponde con el nivel de descomposición más bajo de un proceso actuando sobre un grupo de datos. Por eso, si a un proceso llegan dos grupos de datos, entonces habrá al menos dos ECE's. Se identifica desde una perspectiva puramente funcional[8].
- **Salida Externa de Control (ECX):** Es un único subproceso cuyo cometido es procesar los datos de control que han de salir del sistema, sin realizar lecturas de datos (ya que éstas serán realizadas por el componente ICR). Se corresponde con el nivel de descomposición más bajo de un proceso actuando sobre un grupo de datos. Por eso, si un proceso devuelve dos grupos de datos, entonces habrá al menos dos ECX's. Se identifica desde una perspectiva puramente funcional[8].
- **Lectura Interna de Control (ICR):** Es un único subproceso cuyo cometido es leer datos de control. Se corresponde con el nivel de descomposición más bajo de un proceso actuando sobre un grupo de datos. Por eso, si un proceso lee dos grupos de datos, entonces habrá al menos dos ICR's. Se identifica desde una perspectiva puramente funcional[8].
- **Escritura Interna de Control (ICW):** Es un único subproceso cuyo cometido es escribir datos de control. Se corresponde con el nivel de descomposición más bajo de un proceso actuando sobre un grupo de datos. Por eso, si un proceso escribe sobre dos grupos de datos, entonces habrá al menos dos ICW's. Se identifica desde una perspectiva puramente funcional[8].

El siguiente esquema muestra cómo se relacionan algunos de estos componentes con un proceso de control cualquiera:

[8] Esto significa que el grupo de datos queda perfectamente definido en la especificación de requisitos de la aplicación.

Figura 6. Funciones de Control Transaccional relacionadas con un sistema de control.

Al igual que sucedía en el FPA, todas las nuevas funciones definidas se relacionan con la perspectiva funcional de la aplicación, más que con la perspectiva técnica. La principal diferencia, entonces, entre el análisis FPA y el análisis FFP propuesto son las nuevas funciones incorporadas (ICLG, ECLG, ECE, ECX, ICW e ICR) y que son únicamente válidas para medir datos de control y procesos de control. Los otros tipos de datos y procesos (llamados aquí *datos* y *procesos de gestión*), son contabilizados con el análisis FPA tradicional, según se muestra en la siguiente tabla:

Tipos de funciones de gestión en la técnica FFP:	
Fichero Lógico Interno (ILF)	Existe en FPA, no cambia en FFP.
Fichero Externo de Interfaz (EIF)	Existe en FPA, no cambia en FFP.
Entrada Externa (EI)	Existe en FPA, no cambia en FFP.
Salida Externa (EO)	Existe en FPA, no cambia en FFP.
Consulta Externa (EQ)	Existe en FPA, no cambia en FFP.
Tipos de funciones de control en la técnica FFP:	
Grupo Lógico de Control Interno (ICLG)	Nuevo tipo de función, similar a ILF.
Grupo Lógico de Control Externo (ECLG)	Nuevo tipo de función, similar a EIF.
Entrada Externa de Control (ECE)	Nuevo tipo de función, similar a un subconjunto de EI.
Salida Externa de Control (ECX)	Nuevo tipo de función, similar a un subconjunto de EO/EQ.
Lectura Interna de Control (ICR)	Nuevo tipo de función, similar a un subconjunto de EI/EO/EQ.
Escritura Interna de Control (ICW)	Nuevo tipo de función, similar a un subconjunto de EI.

Tabla 19. Tipos de funciones en la técnica FFP.

Según lo dicho hasta el momento, el número total de Puntos Función Completos sin Ajustar de una aplicación, puede obtenerse con la siguiente ecuación:

$$FFP = FP\ Gestión + FP\ Control = (FPA - Información\ de\ Control) + FP\ Control$$

... donde FP significa *Function Points* (Puntos Función).

7.4.2. Asignación de Puntos Función Completos a los componentes del sistema:

En el punto anterior hemos visto como identificar los distintos tipos de funciones que nos podemos encontrar en un sistema de tiempo real. El siguiente paso será, lógicamente, asignar un número concreto de Puntos Función Completos a cada una de esas funciones.

Para los tipos de función que hemos llamado *de gestión*, y que ya existían previamente en el análisis FPA (ILF, EIF, EI, EO y EQ), el valor asignado será exactamente el mismo que le asigne la técnica FPA (según las reglas establecidas por la IFPUG en su *Manual de Prácticas para el Conteo de Puntos Función* [IFPUG95]).

Para los tipos de función que hemos llamado *de control*, se siguen las siguientes reglas:

A. Grupos Lógicos de Ocurrencia Múltiple.

Tienen la misma estructura que los ILF's y los EIF's del análisis FPA, por lo que son aplicables las reglas de la IFPUG para contabilizarlos como si se tratara de ILF's o EIF's. Es decir, se ha de determinar el número de *Tipos de Datos* (DET's[9]) y *Tipos de Registros* (RET's) involucrados y determinar su complejidad relativa [IFPUG95].

B. Grupos Lógicos de Ocurrencia Única.

El número de Puntos Función Completos asignados dependerá únicamente del número de DET's involucrados. Para los ICLG's, se utiliza la siguiente ecuación:

$$FFP = \left(\frac{nº\ de\ DET's}{5} + 5 \right)^8$$

...debiéndonos quedar con la parte entera del resultado obtenido. Para los ECLG's, la ecuación es:

$$FFP = \left(\frac{nº\ de\ DET's}{5} \right)$$

...debiéndonos quedar también con la parte entera del resultado obtenido.

Estas fórmulas están construidas de tal forma que tratan de mantener muy similar el tamaño de los *grupos lógicos de ocurrencia única* con el tamaño de los ILF's y EIF's del análisis FPA.

C. Tipos de Función de Control Transaccional.

El número de Puntos Función Completos asignados a las funciones de control transaccional (ECE, ECX, ICW e ICR) depende únicamente del número de DET's. Una vez determinado dicho número, se utiliza la siguiente tabla para traducir DET's a Puntos Función Completos:

DET's	de 1 a 19	de 20 a 50	más de 50
Puntos Función Completos	1	2	3

Tabla 20. Números de FFP's asignados a las funciones de control transaccional, dependiendo del número de DET's.

7.4.3. Estimación del esfuerzo de desarrollo:

La estimación del esfuerzo de desarrollo para una aplicación software de tiempo real, utilizando la técnica FFP, es similar a la de la técnica FPA, sólo que con el FFP tenemos más tipos de funciones distintas que identificar. Lo que ocurre es que esta identificación, si cabe, es más sencilla en el FFP que en el FPA, puesto que los *tipos de funciones de control transaccional* están asociadas sólo con un subproceso. En cambio, los tipos de *funciones orientadas a la gestión* pueden contener varios subprocesos y el agruparlos en procesos elementales puede llevar tiempo.

[9] Un DET es un único campo conocido por el usuario, que es introducido, devuelto, leído o escrito, dependiendo del tipo de función que lo trate.

8. Los Puntos Función 3D.

Entre 1989 y 1992, Scott Whitmire [WHIT95] (de la compañía Boeing) desarrolló los llamados Puntos Función 3D para medida del tamaño funcional. Su intención era doble:

- Por una parte, se trataba de simplificar la aplicación de los Puntos Función tradicionales.
- Y por otra, era necesario ampliar el ámbito de aplicación de los Puntos Función a sistemas con elevada complejidad computacional, como sistemas científicos, de tiempo real, etc...

El término "3D" asociado a su nombre hace referencia al hecho de que los Puntos Función 3D consideran las tres *dimensiones* en las que puede proyectarse un sistema software:

a) La dimensión de los **datos** (bastante similar a la considerada por los Puntos Función clásicos).
b) La dimensión **funcional**, que considera las transformaciones de los datos (la complejidad de los algoritmos utilizados para esas transformaciones).
c) La dimensión del **control**, que considera el número de estados-transiciones principales del sistema.

Así concebidos, los Puntos Función 3D parecen ser mucho más completos y potentes que los Puntos Función tradicionales. Sin embargo, presentan un inconveniente: es necesario disponer de una mayor cantidad de información acerca del sistema, sobre todo referente a la complejidad de los algoritmos que se van a implementar y a las posibles transiciones y estados en los que puede encontrarse el sistema. Disponer de esta información no siempre es tan fácil y menos aún durante las primeras etapas del ciclo de vida.

Una de las aplicaciones conocidas de los Puntos Función 3D es la estimación del esfuerzo de desarrollo requerido para implementar *clases* concretas en diseño orientado a objetos. De esta forma, sirven como métrica intermedia entre los Puntos Función tradicionales y las métricas OO.

Con respecto a los dos objetivos que enunciamos al principio, parece ser que no son bien satisfechos por los Puntos Función 3D. Existen dudas de que realmente sean más fáciles de aplicar que los Puntos Función clásicos y de que sean más eficientes en su ámbito de aplicación que otras métricas más habituales, como los Puntos Característica. Lo que sí es cierto es que, en la actualidad, su utilización no está muy difundida a lo largo del mundo.

9. El estándar ISO/IEC 14143 para Medida del Tamaño Funcional.

9.1. Introducción:

A lo largo de los años, y desde que se dio a conocer el Análisis de Puntos Función propuesto por Albrecht, han sido muchos los métodos de medida del software que se han desarrollado, pero siempre sin un acuerdo común sobre los conceptos fundamentales de Medida del Tamaño Funcional (FSM). Esto provocó la aparición de numerosas inconsistencias entre dichos métodos, por lo que se hizo necesaria la elaboración de un estándar apropiado.

En 1992, y con este propósito, cuatro grandes expertos en Puntos Función (Pam Morris (Australia), Paul Goodman (UK), Rob Donnellan y Craig Scates (USA)) se ponen de acuerdo para trabajar conjuntamente en la elaboración de lo que será el primer estándar para Medida del Tamaño Funcional de aplicaciones software. Pero pronto fue necesaria la colaboración de otros renombrados expertos en la materia: Eberhard Rudolph, conocido como uno de los desarrolladores originales de la técnica de FPA; Charles Symons, autor original del método Mark II para Análisis de Puntos Función [SYMO88][SYMO91]; Alain Abran, de la Universidad de Quebec en Montreal, conocido a nivel mundial por ser uno de los desarrolladores de los Puntos Función Completos (Full Function Points); Carol Dekkers, presidente de la IFPUG; Peter Fagg, presidente de la UKSMA; Frank Mazucco, ex-presidente de la IFPUG; etc...

Tras cinco años de intenso trabajo, en 1997 se publica el borrador oficial de la primera parte del estándar ISO para Medida del Tamaño Funcional. Su título es: *"ISO/IEC 14143-1 Tecnología de la Información – Medida del Software: Parte 1. Definición de conceptos para Medida del Tamaño Funcional"*. Su elaboración ha sido llevada a cabo por el comité técnico JTC1/SC7/WG12 de ISO, con la participación de hasta 19 países distintos, y constituye una de las cinco partes en las que se divide el estándar. La publicación definitiva se hizo de forma oficial en la sede de ISO en Génova, el 11 de Junio de 1998.

9.2. El Estandar ISO/IEC 14143:

Como ya se ha comentado antes, el estándar ISO 14143 para Medida del Tamaño Funcional consta de 5 partes bien diferenciadas. En conjunto, proporcionan un marco de referencia con el cual poder decidir cuándo un método concreto es apto para medir el tamaño funcional de aplicaciones software, y cuál es la bondad de los resultados obtenidos. No pretende ser un método concreto de FSM, por lo que tampoco pretende desplazar a ningún otro ya existente. Las cinco partes de que consta son las siguientes:

- ISO/IEC 14143-1: Definición de conceptos.
- ISO/IEC 14143-2: Adaptación de métodos de medida del software al estándar ISO/IEC 14143-1.
- ISO/IEC 14143-3: Verificación de métodos FSM.
- ISO/IEC 14143-4: Modelo de referencia para FSM.
- ISO/IEC 14143-5: Definición de Dominios Funcionales.

En la actualidad, no están completamente desarrolladas todas las partes de este estándar, ni tampoco están consideradas todas ellas como Estándares Internacionales (IS). Con respecto a la Parte 1, se comentó en el punto anterior que fue publicada oficialmente como IS en 1998. La Parte 2 está considerada actualmente como Borrador de Estándar Internacional (DIS) y está a punto de convertirse en IS. Las partes 3, 4 y 5 están consideradas como Documentos o Reportes Técnicos Preliminares y aún no

se encuentran disponibles. Estos documentos no poseen todavía el rigor y el formalismo de un Estándar Internacional, aunque continuamente se van mejorando. Se prevee la publicación como IS dentro de unos dos o tres años.

En los siguientes apartados trataremos más en profundidad cada una de estas partes:

Parte 1: Definición de conceptos para Medida del Tamaño Funcional (FSM).

Esta primera parte define los conceptos fundamentales relacionados con la Medida del Tamaño Funcional y describe los requisitos generales que debe cumplir un método para que sea considerado un método de FSM. Como objetivo principal, esta parte pretende sentar los principios básicos sobre los que se apoya el FSM. Dispone de un Anexo en el que se indican (a título informativo) algunas aplicaciones de los métodos FSM: para calcular ratios de productividad y niveles de calidad, negociar cambios en el alcance de los proyectos, determinar la proporción de requisitos funcionales satisfechos por un paquete software, controlar la productividad en los procesos de desarrollo, operación y mantenimiento, etc...

Puesto que esta es la única parte del estándar que ha sido publicada como IS, y que se encuentra actualmente disponible, se reproduce a continuación una parte del borrador:

Alcance:

Esta parte de ISO/IEC 14143 **NO** proporciona reglas detalladas sobre cómo seleccionar o utilizar un método concreto para medir el tamaño funcional de aplicaciones software, ni cómo usar los resultados derivados de la utilización de un método particular.

Esta parte es aplicable cuando se determina si un método para medir software es un método FSM. Permite valorar si un método concreto se ajusta a las características del FSM. Es útil para usuarios que, relacionados con la medida del tamaño funcional de aplicaciones software, necesitan conocer las bases o criterios aplicables al FSM, usuarios que precisan saber si un método concreto cumple los requisitos del FSM y, en general, todo tipo de personas relacionadas con la adquisición, uso, desarrollo, soporte, mantenimiento y auditoría del software.

Definiciones:

Para los propósitos de esta parte de ISO/IEC 14143, se aplican las siguientes definiciones[10]:

1. Componente Funcional Básico (BFC)
2. Tipos de BFC
3. Límites
4. Método FSM
5. Dominio Funcional
6. Tamaño Funcional
7. Medida del Tamaño Funcional (FSM)
8. Requisitos Funcionales del Usuario
9. Adaptación Local
10. Requisitos de calidad
11. Requisitos técnicos
12. Alcance del FSM
13. Usuario

[10] El significado de estas definiciones puede consultarse en el *Glosario de Términos del Anexo III*.

Características del método FSM:

Un método FSM deberá tener las siguientes características:

a) Estará basado en una representación de los Requisitos Funcionales del Usuario, desde el punto de vista de los usuarios.
b) Podrá aplicarse tan pronto como haya sido definido cualquier Requisito Funcional del Usuario, y mientras éstos estén disponibles.
c) El tamaño funcional se obtendrá de la valoración de los Componentes Funcionales Básicos.

Un método FSM deberá ser tan independiente como sea posible de métodos o tecnologías de desarrollo software particulares, lo cual facilitará un uso más amplio del método FSM.

Características de los Componentes Funcionales Básicos:

Un BFC tendrá las siguientes características:

a) Expresará solo los Requisitos Funcionales del Usuario.
b) No expresará requisitos técnicos.
c) No expresará requisitos de calidad.
d) Será clasificado como un, y sólo un Tipo de BFC.

Características del Tamaño Funcional:

El tamaño funcional tendrá las siguientes características:

a) No se derivará del esfuerzo exigido para desarrollar el software que está siendo medido.
b) No se derivará del esfuerzo exigido para soportar el software que está siendo medido.
c) Será independiente de los métodos usados para desarrollar el software que está siendo medido.
d) Será independiente de los métodos usados para soportar el software que está siendo medido.
e) Será independiente de los componentes físicos del software que está siendo medido.
f) Será independiente de los componentes tecnológicos del software que está siendo medido.

Requisitos del método FSM:

Un método FSM deberá:

a) Definir los atributos de los BFC's.
b) Definir las reglas usadas para evaluar los BFC's.
c) Definir las unidades en las que es expresado el Tamaño Funcional (p. ej.: Puntos Función).
d) Describir el/los dominios funcionales a los que puede aplicarse el método FSM.
e) Describir el tipo de información necesaria para permitir la aplicación del método FSM.
f) Proporcionar guías sobre cómo documentar un caso específico de FSM.
g) Describir los propósitos para los que el método FSM puede usarse mejor, de tal modo que los usuarios del método puedan juzgar su conveniencia para sus propósitos.
h) Declarar su grado de convertibilidad a otros métodos de medida.

Requisitos de valoración de los Componentes Funcionales Básicos:

Un método FSM deberá:

a) Definir los tipos de BFC de que se dispone.
b) Describir cómo identificar los Requisitos Funcionales del Usuario que serán incluidos en el alcance del FSM.
c) Describir cómo identificar los BFC's dentro de los Requisitos Funcionales del Usuario.
d) Definir cómo clasificar los BFC's en Tipos de BFC, si es que hay más de un Tipo de BFC.
e) Definir cómo asignar un valor numérico a un BFC, según el Tipo de BFC al que pertenezca.
f) Definir la relación, si existe, entre el Tipo de BFC y sus límites.
g) Definir las relaciones, si existen, entre los tipos de BFC's existentes.

Designación del Tamaño Funcional:

El método FSM declarará las convenciones a adoptar cuando el informe del Tamaño Funcional sea calificado con:

a) Las unidades del método FSM.
b) El nombre del método FSM.
c) Un indicador de que se ha usado una adaptación local de un método FSM particular.

Proceso para aplicar un método FSM:

Un método FSM incluirá las actividades siguientes, con el fin de determinar el Tamaño Funcional de una aplicación:

a) Determinar el alcance del FSM.
b) Identificar los Requisitos Funcionales del Usuario dentro del alcance del FSM.
c) Identificar los BFC's dentro de los Requisitos Funcionales del Usuario.
d) Clasificar los BFC's en los Tipos de BFC, si es aplicable.
e) Asignar el valor numérico apropiado a cada BFC.
f) Calcular el Tamaño Funcional.

Convenciones para el etiquetado del método FSM:

Un método FSM deberá:

a) Usar un nombre que lo distinga de todos los otros métodos FSM existentes.
b) Incluir un número de versión añadido al nombre del método, que lo distinguirá de todas las otras versiones del método.

Parte 2: Adaptación de métodos de medida del software al estándar ISO/IEC 14143-1.

Se trata de una guía a través de la cual puede comprobarse hasta qué punto un determinado método de medida del software se adapta a los requisitos especificados en la Parte 1 del estándar. Este proceso nos permitirá obtener las ventajas e inconvenientes de distintos métodos de medida y optar así por el que mejor se adapte a las necesidades concretas de cada caso. Los resultados derivados de este proceso pretenden ser objetivos, imparciales, consistentes, repetibles y representantivos de las características del método que se está comprobando.

Parte 3: Verificación de métodos FSM.

Si la Parte 2 del estándar respondía a la pregunta... ¿responde un método a los requisitos especificados en la Parte 1?, la Parte 3 responderá a la siguiente pregunta: ¿es válido este método en el ámbito de aplicación que me interesa?

Aunque son muchos los métodos de medida del software que se utilizan actualmente en todo el mundo, no hay un marco de referencia con el que expresar su efectividad como métodos de FSM. Esta parte del estándar (en conjunción con la Parte 4 que se comenta más adelante) viene a solucionar este problema, permitiendo la evaluación de la validez o utilidad de los métodos FSM.

Con la introducción de este Documento Técnico (no se trata de un IS), se consiguen dos objetivos básicos:

1. Creación de un marco de referencia con el que los usuarios y desarrolladores puedan evaluar las posibilidades de un método FSM concreto.
2. Comparación de ventajas e inconvenientes entre métodos FSM, para poder optar así por el que mejor se adapte a las necesidades concretas de cada momento.

Parte 4: Modelo de referencia para FSM.

Como sabemos, determinados métodos de medida funcional están restringidos a ámbitos de aplicación muy concretos, con lo que su utilidad disminuye considerablemente si se aplican en otros contextos. Véase el caso de los Puntos Función, muy útiles en aplicaciones informáticas de gestión, pero no tan útiles para sistemas en tiempo real, embebidos, etc... Sería necesario, entonces, disponer de un modelo de referencia con el cual poder determinar la efectividad de un método de medida, según el ámbito en el que se aplique. Esta cuarta parte del estándar ISO 14143 trata de solucionar este problema.

Tiene dos objetivos bien definidos:

1. Proporcionar, como referencia, conjuntos de requisitos de usuario para ayudar en la evaluación de la efectividad de un método FSM concreto.
2. Proporcionar un modelo de referencia para determinar la bondad de los resultados obtenidos por un método FSM.

Algunas ventajas que aporta este Documento Técnico son:

- Los desarrolladores de métodos FSM podrán probar sus métodos en los dominios funcionales para los que han sido creados, pudiendo mejorarlos y refinarlos.
- Los usuarios de un método FSM tendrán a su disposición conjuntos de referencia de requisitos funcionales de usuario (de diferentes dominios funcionales), con los que podrán probar dicho método y compararlo con otros.
- Se reducirá el uso inapropiado que se hace de muchos métodos de medida.

Parte 5: Definición de Dominios Funcionales.

Como se comentó en la parte 4, existen muchos métodos de medida funcional del software, pero también muchos ámbitos de aplicación. Sin embargo, no existe acuerdo en cuáles son las características de los requisitos funcionales de usuario, de tal forma que puedan clasificarse en distintos Dominios Funcionales. La Parte 5 del estándar ISO 14143 tratará de dar solución a este problema.

Con la introducción de este Documento Técnico se obtienen las siguientes ventajas:

- Los usuarios de métodos FSM podrán evaluar las características de los requisitos funcionales de usuario, y catalogarlos como pertenecientes a uno o más dominios funcionales.
- Seleccionar el método FSM que resulta ser el más apropiado para medir los requisitos funcionales de usuario en esos dominios funcionales.
- Los desarrolladores de métodos FSM podrán determinar claramente los dominios funcionales en los que resulta efectiva la aplicación de sus métodos.

Anexo I: Referencias Web.

www.ifpug.org
International Function Point Users Group (IFPUG)

Es la agrupación Internacional de Usuarios de Puntos Función. Promueve y defiende la utilización de los Puntos Función como la métrica más importante para la medida del tamaño funcional. Además, establece criterios y reglas concretas para el conteo de Puntos Función.

www.isbsg.org.au
International Software Benchmarking Standards Group (ISBSG)

Asociación sin ánimo de lucro formada por las más populares asociaciones de métricas del software. Ofrece gran cantidad de información sobre métricas aplicadas a proyectos software.

www.uksma.co.uk
The UK Software Metrics Association (UKSMA)

Promueve el uso de métricas del software, en general.

socrates.cit.gu.edu.au/sc7/index_e.html
ISO/IEC JTC1/SC7

El comité técnico de ISO responsable del desarrollo de estándares en el campo de la Ingeniería del Software y responsable, por tanto, de la elaboración del estándar ISO/IEC 14143 sobre Medida del Tamaño Funcional.

www.sei.cmu.edu
Software Engineering Institute (SEI)

Centro de desarrollo e investigación fomentado por el departamento de defensa de los EE.UU.

www.lmagl.qc.ca
Software Engineering Laboratory in Applied Metrics (SELAM)

Proporciona abundante información sobre el análisis FFP.

sunset.usc.edu/index.html
The Center for Software Engineering

Proporciona amplia información sobre el modelo COCOMO II y facilita aplicaciones software para utilización del modelo.

www.totalmetrics.com
Total Metrics

Dispone de muchos recursos referentes a métricas del software, links, foros de discusión, etc...

www.spr.com
Software Productivity Research, Inc (SPR)

La empresa fundada por Capers Jones.

ricis.cl.uh.edu/SE/SWEN5430/resources.html
Software Metrics Resources (University of Houston)

Anexo II: Referencias Bibliográficas.

[Abran *et al,* 1996] Abran, A. & Robillard, P.N. *"Function Point Analysis: An Empirical Study Of Its Measurement Processes"*. IEEE Transactions on Software Engineering, vol. 22, nº 12, pág. 895,909, Diciembre 1996.

[ALBR79] Albrecht, A.J. *"Measuring Applications Development Productivity"*. Proceedings of IBM Application Development Joint. SHARE/GUIDE Symposium, Monterey, CA, 1979, págs. 83-92.

[ALBR83] Albrecht, A.J. *"Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation"*. IEEE Transactions on Software Engineering, Volumen 9, nº 6, 1983, págs. 639-648.

[Albrecht *et al,* 1983] Albrecht, A.J. & Gaffney, J.E. *"Software Function, Source Lines of Code, and Development Effort Prediction"*. IEEE Transactions on Software Engineering, vol. SE-9, nº 6, Noviembre 1983, pág. 639-647.

[Boehm *et al*, 1995] Boehm B.W., Clark B., Horowitz E., Westland C., Madachy R. & Selby R. *"The COCOMO 2.0 Software Cost Estimation Model"*. In Proceedings of the USC Center for Software Engineering's Focused Workshop on COCOMO 2.0, University of Southern California, 1995.

[BOEHM81] Boehm, Barry W. *"Software Engineering Economics"*, Prentice Hall, 1981.

[FENTO97] Fenton, N.E. & Pfleeger, S.L. *"Software Metrics: A Rigorous and Practical Approach"*. Second Edition, International Thomson Computer Press, 1997.

[GAR95] Garmus, D. & Herron, D. *"Measuring the Software Process: A Practical Guide to Functional Measurement"*. Prentice Hall, 1995.

[HALS77] Halstead, M.H. *"Elements of Software Science"*. New York: Elsevier North-Holland, 1977.

[IFPUG95] International Function Point Users Group. *"Counting Practices Manual. Release 4"*. IFPUG, Westerville, Ohio, April 1995.

[JON86] Jones, C. *"Programming Productivity"*, McGraw-Hill, 1986.

[JON96] Jones, C. *"Applied Software Measurement: Assuring Productivity and Quality"*. Segunda Edición, McGraw-Hill, 1996.

[MARCO82] DeMarco, Tom. *"Controlling Software Projects – Management, Measurement ans Estimation"*. Englewood Cliffs, N.J.: Prentice Hall, 1982.

[PRES97] Pressman, R.S.
"Ingeniería del Software: Un enfoque práctico". Cuarta Edición, McGraw Hill, 1997

[Serge *et al*, 1997] Oligny S., Bourque O. & Abran A.
"An Empirical Assessment of Project Duration Models in Software Engineering". Proceedings of the Eighth European Software Control and Metrics Conference (ESCOM'97).

[SYMO88] Symons, C.R.
"Function Points Analysis: Difficulties and Improvements". IEEE Transactions on Software Engineering, Volumen 14, nº 1, Enero 1988, págs. 2-11.

[SYMO91] Symons, C.R.
"Software Sizing and Estimating: MkII FPA", John Wiley & Sons, New York, 1991.

[WHIT95] Whitmire, S.A.
"An Introduction to 3D Function Points", Software Development, vol. 3, nº 4, Abril 1995.

Anexo III: Glosario de Términos y Acrónimos.

Acrónimo	Término	Definición
	Adaptación local.	Es un método de FSM que se ha modificado para uso local. Podría producir tamaños funcionales diferentes de aquéllos obtenidos antes de la modificación.
	Alcance del FSM.	Es el conjunto de Requisitos Funcionales del Usuario que se incluyen en una instancia específica del FSM.
BFC:	**Basic Functional Component.**	Componente Funcional Básico: Es la unidad elemental de los Requisitos Funcionales del Usuario (UFR) definida y usada por un Método FSM para los propósitos de las métricas.
	Componente elemental.	En el contexto de aplicación de la métrica *Bang*, es cada uno de los 6 elementos básicos que se obtienen de la división y descomposición del modelo de especificación: *Primitivas funcionales, Datos elementales, Objetos, Interrelaciones, Estados y Transiciones*.
COCOMO II	**Constructive Cost Model II**	Modelo Constructivo del Coste II: Modelo resultante de la adaptación del modelo COCOMO a las características y necesidades de los procesos de desarrollo software actuales.
COCOMO	**Constructive Cost Model.**	Modelo Constructivo del Coste: Modelo desarrollado por Boehm para la estimación del esfuerzo, duración y coste (entre otros factores) de proyectos software.
	Consulta Externa.	En el ámbito de aplicación del análisis FPA, es un proceso elemental con componentes de entrada y de salida que consiste en la selección y recuperación de datos de uno o más *Ficheros Lógicos Internos* o de uno o más *Ficheros Externos de Interfaz*, y su posterior devolución al usuario o aplicación que los solicitó.
CFPI	**Corrected Functional Primitive Increment.**	En el contexto de aplicación de la métrica *Bang*, el CFPI representa el tamaño corregido de una primitiva funcional. Su valor es: $CFPI_i = TC_i * log_2 (TC_i)$
CFP	**Corrected Functional Primitive.**	En el contexto de aplicación de la métrica *Bang*, el CFP representa el tamaño funcional total de la aplicación medida. Su valor es: $CFP = \Sigma CFPI_i$
COB	**Corrected Object**	En el contexto de aplicación de la métrica *Bang*, el COB representa el tamaño funcional total de la aplicación medida. Su valor es: $COB = \Sigma COBI_i$
COBI	**Corrected Object Increment.**	En el contexto de aplicación de la métrica *Bang*, el COBI representa el factor de corrección (peso) de un objeto del modelo de datos del sistema, en función de sus interrelaciones con otros objetos.

Acrónimo	Término	Definición
DEI	**Data Element Input.**	En el contexto de aplicación de la métrica *Bang*, es un dato elemental de entrada al sistema.
DER	**Data Element Retained.**	En el contexto de aplicación de la métrica *Bang*, es un dato almacenado dentro del sistema.
DET	**Data Element Type.**	Tipo de Dato: Es un único campo conocido por el usuario, que es introducido, devuelto, leído o escrito, dependiendo del tipo de función que lo trate.
DE	**Data Element.**	Véase *Dato Elemental*.
DO	**Data Output**	En el contexto de aplicación de la métrica *Bang*, es un dato de salida del sistema.
	Dato elemental.	En el contexto de aplicación de la métrica *Bang*, es un dato concreto e indivisible que forma parte del *diccionario de datos* del modelo funcional.
	Dominio Funcional.	Es un tipo de software basado en los Requisitos Funcionales del Usuario, que son adecuados al FSM.
DIS	**Draft of International Standard.**	Borrador de Estándar Internacional.
	Entrada Externa.	En el ámbito de aplicación del análisis FPA, es un proceso elemental a través del cual se permite la entrada de datos al sistema.
	Estado.	En el contexto de aplicación de la métrica *Bang*, es el componente elemental que se deriva de sucesivas descomposiciones del modelo de comportamiento (descomposiciones de un *diagrama de estados/transiciones*, por ejemplo).
ECE	**External Control Entry.**	Entrada Externa de Control: En el ámbito de aplicación del análisis FFP, es un único subproceso cuyo cometido es procesar los datos de control que vienen desde fuera del sistema, sin actualizar datos.
ECX	**External Control Exit.**	Salida Externa de Control: En el ámbito de aplicación del análisis FFP, es un único subproceso cuyo cometido es procesar los datos de control que han de salir del sistema, sin realizar lecturas de datos.
ECLG	**External Control Logical Group.**	Grupo Lógico de Control Externo: En el ámbito de aplicación del análisis FFP, es un grupo de datos de control lógicamente relacionados, y usados (pero no actualizados) por la aplicación que se está midiendo.
EI	**External Input.**	Véase *Entrada Externa.*

Acrónimo	Término	Definición
EIF	**External Interface File.**	Véase *Fichero Externo de Interfaz.*
EO	**External Output.**	Véase *Salida Externa.*
EQ	**External Query.**	Véase *Consulta Externa.*
	Fichero Externo de Interfaz.	En el ámbito de aplicación del análisis FPA, es un conjunto de datos definidos por el usuario, que están relacionados lógicamente, y que sólo son usados para propósitos de referencia. Los datos residen en su totalidad fuera de los límites de la aplicación y son mantenidos por otras aplicaciones.
	Fichero Lógico Interno.	En el ámbito de aplicación del análisis FPA, es un conjunto de datos definidos por el usuario y relacionados lógicamente, que residen en su totalidad dentro de la propia aplicación, y que son mantenidos a través de la *Entradas Externas* del sistema.
FFP	**Full Function Points.**	Puntos Función Completos: Es una métrica del tamaño funcional, de probada eficiencia en la medición de sistemas de control, tiempo real y embebidos.
FPA:	**Function Point Analisys.**	Análisis de Puntos Función: Es un método de medida del tamaño funcional de aplicaciones software, gereneralmente orientadas a la gestión.
FPM	**Functional Primitive Modified.**	En el contexto de aplicación de la métrica *Bang*, son primitivas funcionales externas al sistema que deben modificarse para adaptarlas al nuevo sistema.
FP	**Functional Primitive.**	Véase *Primitiva Funcional*.
FSM:	**Functional Size Measurement.**	Medida del Tamaño Funcional: Es el proceso de medir el Tamaño Funcional de una aplicación software.
GSC	**General System Characteristics.**	Características Generales del Sistema: Cada una de las 14 que se utilizan en el análisis FPA (ver tabla 2).
ICLG	**Internal Control Logical Group.**	Grupo Lógico de Control Interno: En el ámbito de aplicación del análisis FFP, es un grupo de datos de control lógicamente relacionados, y actualizados por la aplicación que se está midiendo.
ICR	**Internal Control Read.**	Lectura Interna de Control: En el ámbito de aplicación del análisis FFP, es un único subproceso cuyo cometido es leer datos de control.
ICW	**Internal Control Write.**	Escritura Interna de Control: En el ámbito de aplicación del análisis FFP, es un único subproceso cuyo cometido es escribir datos de control.
ILF	**Internal Logical File.**	Véase *Fichero Lógico Interno.*

Acrónimo	Término	Definición
IFPUG:	**International Function Point Users Group.**	Agrupación Internacional de Usuarios de Puntos Función: Promueve y defiende la utilización de los Puntos Función como la métrica más importante para la medida del tamaño funcional. Además, establece criterios y reglas concretas para el conteo de Puntos Función.
ISBSG:	**International Software Benchmarking Standards Group.**	Asociación sin ánimo de lucro formada por las más populares asociaciones de métricas del software. Ofrece gran cantidad de información sobre métricas aplicadas a proyectos software.
IS	**International Standard.**	Estándar Internacional.
ISO:	**International Standards Organization.**	Organización Internacional para la Estandarización.
	Interrelación.	En el contexto de aplicación de la métrica *Bang*, es el componente elemental derivado de las relaciones (conexiones) existentes entre unos objetos y otros del modelo de datos.
KLDC	**Kilo-Líneas de Código Fuente.**	1000 líneas del código fuente de una aplicación software.
KSLOC	**Kilo-Source Line Of Code.**	Véase *Kilo-líneas de código fuente.*
	Límites.	Un límite es una interfaz conceptual entre el software estudiado y sus usuarios.
LOC	**Line Of Code.**	Véase *Línea de código.*
LDC	**Línea de Código.**	Cada una de las líneas del código fuente de una aplicación software.
	Método FSM.	Es una implementación específica del FSM, definida por un conjunto de reglas acordes con las características definidas en la Parte 1 del estándar ISO/IEC 14143.
NOP	**New Objet Points.**	Es el número total de Puntos Objeto Ajustados.
OB	**Objetc.**	Véase *Objeto*.
	Objeto.	En el contexto de aplicación de la métrica *Bang*, es el componente elemental que se deriva de sucesivas descomposiciones del modelo de datos (descomposiciones de un *diagrama E/R*, por ejemplo).
	Primitiva funcional.	En el contexto de aplicación de la métrica *Bang*, es el componente elemental que se deriva de sucesivas descomposiciones del modelo funcional (descomposiciones de los procesos de un *diagrama de flujo de datos*, por ejemplo).
	Punto Característica.	Es la métrica del tamaño funcional de una aplicación, propuesta inicialmente por Capers Jones en 1986.

Acrónimo	Término	Definición
	Punto Función.	También llamado *Punto de funcionalidad*, es la métrica del tamaño funcional de una aplicación, propuesta inicialmente por Albrecht en 1979.
	Punto Objeto.	Es la métrica del tamaño funcional de una aplicación, propuesta inicialmente por Boehm en su modelo de estimación COCOMO II.
RAD	**Rapid Application Development.**	Desarrollo Rápido de Aplicaciones: Son sistemas capaces de generar automáticamente, y con el mínimo esfuerzo, componentes software utilizables en la aplicación que se está desarrolando (interfaces gráficos, prototipos operativos, etc...)
RE	**Relationship.**	Véase *Interrelación*.
	Requisito de calidad.	Cualquier requisito relativo a la calidad del software, según se define en el estándar ISO 9126:1991
	Requisito técnico.	Cualquier requisito que relaciona la tecnología y el entorno, para el desarrollo, mantenimiento, soporte y explotación del software.
	Salida Externa.	En el ámbito de aplicación del análisis FPA, es un proceso elemental a través del cual se permite la salida de datos del sistema.
SPR:	**Software Productivity Research, Inc.**	La empresa fundada por Capers Jones.
SLOC	**Source Line Of Code.**	Véase *Línea de código.*
ST	**State.**	Véase *Estado*.
	Tamaño Funcional.	Es el tamaño de la aplicación software, derivado de la cuantificación de los Requisitos Funcionales del Usuario.
	Tipo de software.	Un tipo de software es una categoría definida de software. Por ejemplo, puede haber software para sistemas de tiempo real, software para sistemas empotrados, software para control de procesos industriales, etc...
	Tipos de BFC.	Un tipo de BFC es una categoría definida de BFC's.
	Token.	En el contexto de aplicación de la métrica *Bang*, es un dato o conjunto de datos que la primitiva funcional trata como un todo, es decir, no requiere ser dividido para que la primitiva funcional pueda utilizarlo.
	Transición.	En el contexto de aplicación de la métrica *Bang*, es la acción que hace pasar a un sistema de un estado a otro del modelo de comportamiento

Acrónimo	Término	Definición
TR	**Transition.**	Véase *Transición*.
UKSMA:	**United Kingdom Software Metrics Association.**	Promueve el uso de métricas del software, en general.
UFR:	**User Functional Requirements.**	Requisitos Funcionales del Usuario: Es un subconjunto de los requisitos de usuario. Los UFR's representan las prácticas y los procedimientos de usuario que el software debe realizar para completar las necesidades de los usuarios. Se excluyen los requisistos de calidad y cualquier requisito técnico.
	Usuario	Es cualquier persona que especifica Requisitos Funcionales de Usuario y/o cualquier persona o cosa que se comunica o interactúa con el software en cualquier momento.
VAF:	**Value Adjustment Factor.**	Factor de Ajuste: También llamado Factor de Complejidad Técnica.