2.2 Estimación de tamaño: puntos de función

FUENTES:
Análisis y diseño de aplicaciones informáticas de gestión (cap. 5) de Piattini M. G., Calvo-Manzano, J.A., Cervera, J. y Fernández, L., editado por Ra-Ma, 2004.
Planificación y gestión de proyectos informáticos, de José Antonio Gutiérrez de Mesa y Carmen Pagés Arévalo, editado por Servicio de publicaciones de la UAH, 2005.

PUNTOS DE FUNCIÓN

El método de estimación de costes mediante los puntos de función ha sido denominado FPA o Análisis de Puntos de Función. Este método se basa en una **métrica que cuantifica la funcionalidad que hay que entregar al usuario al construir una aplicación**. Dicha métrica se denomina PF o puntos de función (FP: Function Points). La propuesta inicial de los puntos de función fue realizada por A. J. Albrecht en 1979 y, desde entonces, ha sufrido diversos refinamientos posteriores. Todas las variedades de puntos de función se apoyan en datos que implican, preferentemente, la existencia de una especificación más o menos formalizada. El modelo inicial es la estandarización IFPUG (International Function Point User Group: http://www.ifpug.org/).

El proceso es el que se muestra en la Ilustración 1: el cálculo de los PF no ajustados se realiza contando elementos externos de la aplicación con pesos (entradas, salidas, consultas y ficheros maestros) y a continuación se ajustan los PF haciendo un ajuste de complejidad según el grado de influencia de 14 factores.

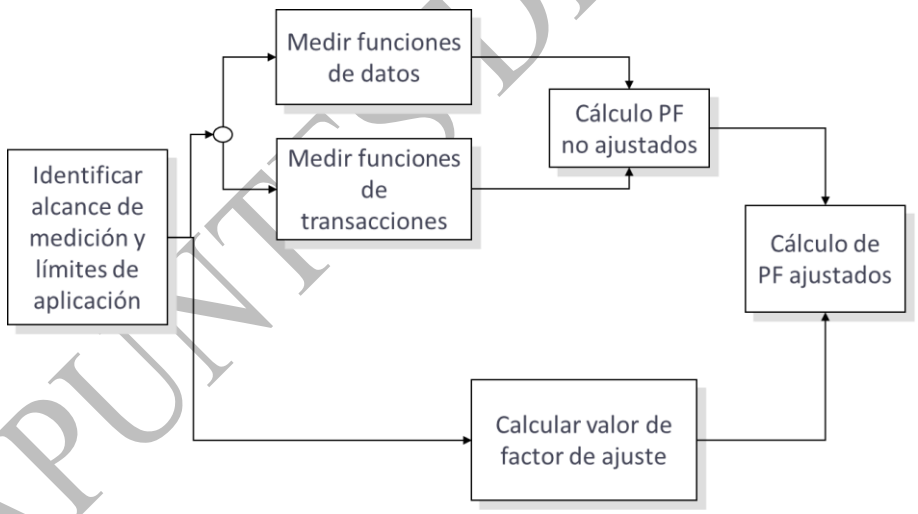

Ilustración 1. Proceso de medición de los puntos de función

1. **Identificar alcance de medición y límites de aplicación.**

Establecer los límites dependerá del objetivo de la aplicación del método (ver la Ilustración 2). Si estamos en un proceso de mejora o mantenimiento de una aplicación se tendrán en cuenta las funciones que se añaden, cambian o borran y su impacto en los datos. Si estamos en desarrollo de una nueva aplicación tendremos en cuenta todas las funciones del proyecto. Si queremos medir una aplicación terminada, según el propósito, podremos medir todas las funciones entregadas o las funciones que utiliza un departamento o usuario, por ejemplo.

La identificación de los límites de la aplicación debe estar basada en la visión de usuario, no en la de los técnicos, y debe tener en cuenta las aplicaciones relacionadas según la funcionalidad de negocio. Una aplicación relacionada es aquella que contiene información que se comparte con la aplicación que medimos, pero esa información no es mantenida (altas, bajas consultas y modificaciones directas) por la aplicación que medimos, sino por la relacionada.

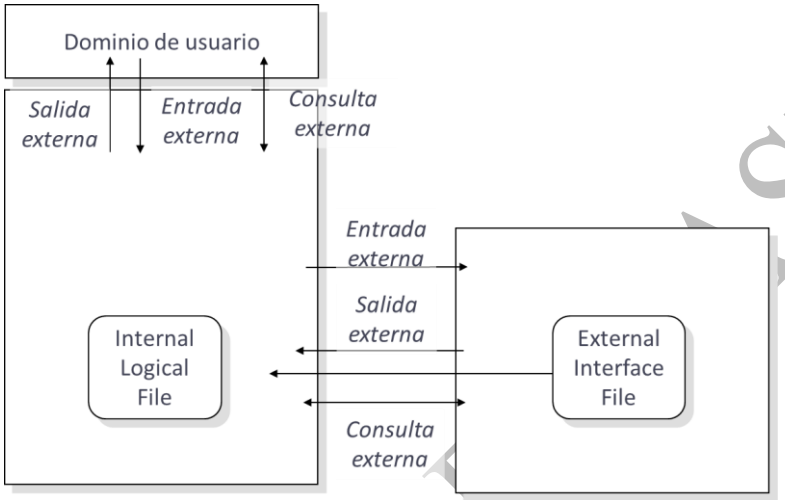

Ilustración 2. Esquema para determinar los límites del sistema y los elementos para los puntos de función

2. **Medir funciones de datos.**

Las funciones de datos son de dos tipos:

- **Ficheros Lógicos Internos** (*Internal Logical Files* o ILF): Es un grupo de datos relacionados, tal como los percibe el usuario y que son mantenidos por la aplicación. Los ficheros se cuentan una sola vez, independientemente del número de procesos que acceden a ellos.
- **Ficheros de Interfaz Externos** (*External Interface Files* o EIF): Es un grupo de datos relacionados, tal como los percibe el usuario, referenciados por la aplicación y que son mantenidos por otra aplicación. Son ficheros internos de otra aplicación.

Es importante asumir la perspectiva del usuario (lógica) frente a la implementación (física): un fichero físico no tiene por qué equivaler a uno lógico (aunque la tabla relacional o el fichero planos se aproximan).

Para calcular la complejidad de cada función de datos se deben contar los **datos elementales (DET)** y los **registros elementales (RET)**.

Un **DET** es un único campo no recursivo del ILF o EIF que es reconocible por el usuario. Reglas para DET:

- Contar un DET por cada campo único y reconocible por el usuario de un ILF o EIF
- Ejemplo: un número o un dato físicamente almacenado en múltiples campos se cuenta como un solo DET.
- Contar un DET por cada unidad de dato en un ILF o EIF que existe. Por ejemplo en una aplicación de RRHH, se mantiene información de empleados en un ILF. Se cuenta un DET por cada campo del ILF: nombre, apellido, fecha nacimiento, puesto, etc. Campos repetitivos idénticos en formato para múltiples ejemplares de valores (no hay 1ªFN) se cuentan como uno, por ejemplo si hay 12 campos para cantidades mensuales y un total anual, se cuenta como 2 DET.

Un **RET** es un subgrupo de elementos de datos en un ILF o EIF que es reconocible por el usuario. El subgrupo puede ser opcional, el usuario tiene opción de usar uno o ninguno durante el proceso elemental de crear o añadir, u obligatorio, el usuario debe usar al menos uno. Por ejemplo en una aplicación de RRHH, aparte de la información general del empleado, la información debe indicar si es asalariado o autónomo (cada uno con distintos campos) y puede tener información o no sobre familiares. La información general es obligatoria y el resto (asalariado, autónomo, familiares) es opcional. Reglas para RET:

- Contar un RET por cada subgrupo opcional u obligatorio de un ILF o EIF.
- Si no hay subgrupos, contar el ILF o EIF como un RET.

El proceso completo de conteo de funciones de datos es el siguiente:

1 Identificar grupos lógicos de datos e información de control y determinar si cada grupo se cuenta como ILF o EIF.
2 Aplicar las reglas de cuenta de DET y RET
3 Clasificar la complejidad de cada ILF/EIF según la basada en las cantidades de DET y RET (ver Tabla 1).

Complejidad	1 a 19 DET	20 a 50 DET	≥ 51 DET
1 RET	Baja	Baja	Media
2-5 RET	Baja	Media	Alta
≥ 6 RET	Media	Alta	Alta

Tabla 1. Clasificación de complejidad de ILF y EIF según los DET y RET

4 Buscar los pesos aplicables a cada ILF o EIF en función de su complejidad y calcular aportación para el cálculo de PF no ajustados (ver Tabla 2).

Tipo	Baja	Media	Alta
ILF	x7	x10	x15
EIF	x5	x7	x10

Tabla 2. Pesos según complejidad para ILF y EIF

3. Medir funciones de transacciones.

Las funciones de transacciones representan la funcionalidad proporcionada al usuario por la aplicación para el procesamiento de datos. Son de tres tipos:

Entradas (External Input o EI): Son todos aquellos procesos que hacen llegar datos a la aplicación desde el exterior, desde un usuario u otra aplicación. El flujo de datos deberá tener una sola dirección, del exterior al interior. Como consecuencia de una entrada, siempre deberá actualizarse un fichero lógico interno. Ejemplos: pantallas de entrada de datos, lector de códigos de barras, lector de tarjetas magnéticas y electrónicas, captura de imágenes, voz, etc.
Salidas (External Output o EO): Son todos aquellos procesos que hacen llegar datos desde la aplicación hacia el exterior, a un usuario o a otra aplicación. El flujo de datos deberá tener una sola dirección, del interior al exterior. Ejemplos: pantallas de salida de datos, listados, grabación de bandas magnéticas, transferencia de datos a otras aplicaciones, ya sea mediante ficheros o transmisión de datos, etc.
Consultas (External Inquiry o EQ): Son todos aquellos procesos que están formados por una combinación de entradas y salidas, produciendo una consulta a los datos. El flujo de datos deberá tener dos direcciones. Como consecuencia de una consulta no se modifican los datos del sistema. La complejidad de la consulta viene dada por la mayor entre la entrada y la salida. La entrada y la salida de una consulta se valoran por separado y la consulta se valora con la complejidad mayor de las dos obtenidas. Las consultas con modificación de datos se consideran dos elementos: una entrada y una salida. Estos dos elementos se valoran y cuentan por separado.

Para calcular la complejidad de una función de transacciones se cuentan los datos elementales (DET), con el mismo significado que en las funciones de datos, y los ficheros referenciados (FTR), los ficheros lógicos internos o externos que hay que escribir, actualizar o simplemente consultar para realizar la función.

En las EI se cuenta un DET por cada campo no repetitivo introducido desde el exterior (usuario, lector magnético, etc.) y por cada campo no introducido desde el exterior pero que se mantiene en la realización de la transacción. Por ejemplo, al dar de alta un empleado, se cuentan los campos introducidos (nombre, apellidos, fecha de nacimiento, etc.) y el código de empleado generado por la aplicación. También se cuentan campos que indican que ha ocurrido un error durante el proceso o que confirman que el proceso se ha completado. Por ejemplo, si un usuario intenta añadir un empleado existente, el sistema genera varios mensajes de error y el campo incorrecto se resalta, hay que contar un solo DET para todas las respuestas de error del sistema. También se cuenta un solo DET para especificar la acción a tomar por la entrada externa, es decir, la tecla de función o la opción elegida para ejecutar la transacción..

En las EO se cuenta un DET por cada campo no recursivo que aparece en la salida (listado, pantalla, grabación, etc.). No se cuentan los literales (títulos de informes) ni las marcas de paginación. Se cuenta un único DET para los campos que el usuario ve como

uno aunque estén almacenados por partes (por ejemplo una fecha, aunque se almacene como tres campos: día, mes y año). También se cuenta un DET por los posibles mensajes de error de la transacción.

El proceso completo de cuenta de funciones de transacciones es el siguiente:

1. Identificar las transacciones que obtienen información del exterior y actualizan datos (entradas), dan información al exterior (salidas) u obtienen y dan información al exterior sin actualizar datos (consultas).
2. Para cada una de ellas aplicar las reglas de cuenta de DET y FTR
3. Clasificar la complejidad de cada elemento según las siguientes tablas basada en las cantidades de DET y FTR (ver Tabla 3 y Tabla 4).

EI	1 a 4 DET	5 a 15 DET	≥ 16 DET
0-1 FTR	Baja	Baja	Media
2 FTR	Baja	Media	Alta
≥ 3 FTR	Media	Alta	Alta

Tabla 3. Clasificación de complejidad de EI según los DET y FTR

EO	1 a 4 DET	5 a 15 DET	≥ 16 DET
0-1 FTR	Baja	Baja	Media
2-3 FTR	Baja	Media	Alta
≥ 4 FTR	Media	Alta	Alta

Tabla 4. Clasificación de complejidad de EO según los DET y FTR

4. Buscar los pesos aplicables a cada elemento en función de su complejidad y calcular aportación para el cálculo de PF no ajustados (ver Tabla 5).

Tipo	Baja	Media	Alta
EI	x3	x4	x6
EO	x4	x5	x7

Tabla 5. Pesos según complejidad para EI y EO

5. **Cálculo de puntos de función no ajustados (PFNA).**

Una vez medidas todos los elementos de función se rellena la Tabla 6 de forma que se suman los puntos de función obtenidos en total.

	Complejidad baja	Complejidad media	Complejidad alta	Total fila
ILF	x 7	x 10	x 15	
EIF	x 5	x 7	x 10	
EI	x 3	x 4	x 6	
EO	x 4	x 5	x 7	
EQ	x 3	x 4	x 6	
Puntos de función no ajustados				

Tabla 6. Tabla de cálculo de los puntos de función no ajustados

6. Calcular valor de factores de ajuste.

Existen 14 factores que contribuyen a la complejidad de una aplicación. Se debe valorar cada uno de ellos dentro de una escala del cero al cinco, en función de las características del sistema que se va a construir. El valor de los factores de ajuste (VFA) permite modificar los PF no ajustados en un 35% para producir los PF finales o ajustados. Cada factor de complejidad, pudiendo oscilar entre 0 y 5, afecta en un +/- 2,5% en los PFNA, por tanto cada punto de factor (de 0 a 5) es un 1% del total.

La escala de cada factor representa lo siguiente:

0 = No presente o sin influencia
1 = Influencia ocasional
2 = Influencia moderada
3 = Influencia media
4 = Influencia significativa
5 = Influencia muy fuerte

Los factores son los siguientes:

1. Comunicación de Datos. Los datos usados en el sistema se envían o reciben por líneas de comunicaciones.

2. Proceso Distribuido. Existen procesos o datos distribuidos y el control de éstos forma parte del sistema.

3. Rendimiento. Si el rendimiento es un requisito del sistema, es decir, es crítico algún factor como tiempo de respuesta o cantidad de operaciones por hora. Se tendrá que hacer consideraciones especiales durante el diseño, codificación y mantenimiento.

4. Configuración operacional compartida. El sistema tendrá que ejecutarse en un equipo en el que coexistirá con otros, compitiendo por los recursos, teniendo que tenerse en cuenta en la fase de diseño.

5. Ratio de Transacciones. La tasa de transacciones será elevada. Se tendrán que hacer consideraciones especiales durante el diseño, codificación e instalación.

6. Entrada de Datos On-line. La entrada de datos será directa desde el usuario a la aplicación, de forma interactiva.

7. Eficiencia para el Usuario Final. Se demanda eficiencia para el trabajo del usuario, es decir, se tiene que diseñar e implementar la aplicación con interfaces fáciles de usar y con ayudas integradas.

8. Actualizaciones On-line. Los ficheros maestros y/o las Bases de Datos son modificados de forma interactiva.

9. Complejidad del Proceso Interno. La complejidad interna en un proceso está en función de las siguientes características:
   - Algoritmos matemáticos complejos.
   - Proceso con lógica compleja.
   - Muchas excepciones, consecuencia de transacciones incompletas, que deberán tratarse.
   - Manejar múltiples dispositivos de entrada / salida.
   - Se incorporarán sistemas de seguridad y control.

10. Reusabilidad del código. Es necesario hacer consideraciones especiales durante el diseño, codificación y mantenimiento para que el código se reutilice en otras aplicaciones.

11. Contempla la conversión e instalación. Se proveerán facilidades de conversión del sistema antiguo durante la puesta en marcha del nuevo.

12. Facilidad de Operación. Facilitar la explotación real de la aplicación, dedicando especial atención durante el diseño, codificación y pruebas del sistema. Se pueden tener en cuenta las siguientes posibilidades de automatización:
   - Procesos de arranque, back-up y recuperación pero con o sin intervención del operador.
   - Manejo de cintas u otros dispositivos manual o automático.

13. Instalaciones Múltiples. El sistema ha de incluir los requerimientos de diversas empresas o departamentos en donde se ejecutara (incluso plataformas).

14. Facilidad de Cambios. Se tendrá que hacer consideraciones especiales para que en el sistema sea fácil de introducir cambios y fácil de adaptar al usuario.

Los detalles de cómo valorar cada factor deben consultarse en los manuales detallados de cálculo con puntos de función. La fórmula para el cálculo del VFA es:

VAF = (FCT * 0.01) + 0.65

Donde FCT es la suma del valor dado a cada uno de los factores. Por ejemplo si damos valor 0 a todos los factores, excepto 3 a Rendimiento y 5 a Facilidad de cambio, VFA sería 3+5=15.

7. **Cálculo de puntos de función ajustados.**

Finalmente se calculan los PFA como los PFNA multiplicados por el VFA.

Backfiring

Los puntos de función ajustados nos dan una medida de tamaño del software. Muchas organizaciones y empresas han estudiado la equivalencia de los puntos de función con las LOC. La idea es poder estimar los puntos de función en aplicaciones ya desarrolladas de las que no existe una especificación detallada y actualizada o cuya cuenta de puntos de función requiere bastante esfuerzo sin tener recursos para realizarla. A este proceso se le llama *backfiring*. De esta forma, se plantea estimar los puntos de función a través de las LOC del sistema terminado que sí están disponibles a través de herramientas.

Normalmente, en la realidad, no tiene sentido usar la equivalencia entre puntos de función y LOC para pasar a estimar esfuerzo y plazo de tiempo para un proyecto: es decir, una vez calculados los puntos de función lo habitual es usar datos históricos de productividad en la organización para estimar el esfuerzo y los plazos. También se han desarrollado ecuaciones de costes que toman directamente como parámetro los puntos de función (por ejemplo, el modelo de Symons). Por el contrario, transformar los puntos de función a LOC para luego usar este número como parámetro para modelos de estimación como COCOMO no tiene sentido y supone riesgos como considerar factores de ajuste dos veces: una vez con los puntos de función y otra con el modelo de costes (por ejemplo, los factores de COCOMO). Por supuesto, a efectos de ejercicios educativos sí se puede plantear esta conexión de puntos de función a LOC para ejercitar el uso de modelos.

La lista siguiente muestra algunos ejemplos según el entorno de programación utilizado:

- Assembler: 213
- C básico: 128
- FORTRAN 77: 105
- ANSI COBOL 85: 91
- Pascal: 91
- PL/I: 80
- Modula 2: 80
- Ada: 71
- Prolog: 64
- ANSI/Turbo BASIC: 64

- C++: 53
- Base de datos: 40
- OO por defecto: 29
- Query: 13

Otra utilidad de los puntos de función es la evaluación de la productividad del personal en cualquier etapa de desarrollo. Así, se define la productividad como el cociente entre los puntos de función implementados y los recursos consumidos en su desarrollo. Esta forma de medir está adquiriendo una aceptación cada vez mayor. El uso de los puntos de función ha sido tradicionalmente amplio en los EE.UU. y en el Reino Unido pero se ha extendido ya bastante a muchos otros países.

La Tabla 7 es un ejemplo de datos obtenidos a partir de un registro de datos históricos.

Nombre Proyecto	Puntos de Función	Lenguaje	Esfuerzo en horas	Horas/PF
Sénia	200	COBOL	5.017	25
Paláncia	150	PASCAL	2.569	17
Turia	375	4GL	3.011	8
Albufera	500	PASCAL	9.479	19
Magro	425	4GL	3.342	8
Cabriel	800	PASCAL	13.349	17
Júcar	180	PASCAL	2.800	16
Serpis	325	4GL	2.541	8
Montnegre	225	PASCAL	4.528	20
Segura	470	COBOL	13.218	28

Tabla 7. Ejemplo de datos históricos de proyectos

EJEMPLO DE PUNTOS DE FUNCIÓN

Calcula los puntos de función de una aplicación que consta de los siguientes elementos:

- Ficheros propios: Proveedores (25 campos) y Direcciones (10 campos);
- Ficheros externos: Histórico (20 campos, de los cuales la aplicación accede a 6);
- Programa batch de Carga. Programa que carga completos los ficheros de Proveedores y Direcciones
- Programa on-line de Consulta. Programa que lee de pantalla el identificativo de un proveedor. Si este no existe da un mensaje de error y si existe muestra por pantalla 12 campos del fichero Proveedores y 5 del fichero Histórico;
- Programa on-line de Modificación. Programa que modifica la dirección de los proveedores, para ello lee de pantalla el identificativo del proveedor y nueve campos del fichero de Direcciones. Si el proveedor no existe en el fichero de Direcciones da un mensaje de error y si existe modifica los nueve campos en el fichero de Direcciones

Suponemos que los factores de complejidad a utilizar son objetivo de rendimiento (FC3) medio, integración de la aplicación (FC4) bajo, facilidad de cambios (FC14) alto y entrada de datos on-line (FC6) muy alto, el resto tiene valor despreciable (0). Sabiendo que históricamente se han necesitado 11 horas por punto de función, hay que calcular el esfuerzo.

Archivos lógicos internos:

NOMBRE	NÚM.REGISTROS LÓGICOS	ATRIBUTOS	CLASIFICACION
Proveedores	1	25	BAJA
Direcciones	1	10	BAJA

Archivos de interfaz externa

NOMBRE	NÚM.REGISTROS LÓGICOS	ATRIBUTOS	CLASIFICACION
Histórico	1	6	BAJA

Entradas

NOMBRE	NÚM.FICHEROS ACCEDIDOS	ATRIBUTOS	CLASIFICACION
Carga	2	35	ALTA
Modificación	1	10	BAJA

Salidas

NOMBRE	NÚM.FICHEROS ACCEDIDOS	ATRIBUTOS	CLASIFICACION
Modificación	1	10	BAJA

Consultas

NOMBRE	NÚM.FICHEROS ACCEDIDOS	ATRIBUTOS	CLASIFICACION
Consulta (como entrada)	2	1	BAJA
Consulta (como salida)	2	18	MEDIA

	BAJA	MEDIA	ALTA	TOTAL
ENTRADAS	1 * 3	0 * 4	1 * 6	9
SALIDAS	1 * 4	0 * 5	0 * 7	4
CONSULTAS	0 * 3	1 * 4	0 * 6	4
FICH. LOGICOS	2 * 7	0 * 10	0 * 15	14
FICH. INTERFACES	1 * 5	0 * 7	0 * 10	5

Total puntos de función sin ajustar = 9 + 4 + 4 + 14 + 5 = 36

Total puntos de función ajustados = 36*(0,65+(0,01*(3+2+4+5))) = 36*(0,65+0,14) = 36*0,79 = 28,44

Históricamente se han necesitado 2 horas por punto de función. Para esta aplicación el esfuerzo necesario será 28,44 * 11 = 312,84 horas.

MARK II.

Existen variaciones al método inicial de puntos de función. Mark II es un ejemplo.

Consiste en contar para cada transacción lógica:

Elementos de datos de entrada: NI
Entidades referenciadas: NE
Elementos de datos de salida: NO

Los puntos de función no ajustados se calculan con esta fórmula:

$$PNA = WI \cdot NI + WE \cdot NE + WO \cdot NO$$

Se deberían calibrar los pesos (WI, WE y WO) para cada entorno de programación. Por ejemplo, los pesos medios obtenidos por Nolan Norton son:

WI = 0,58; WE = 1,66; WO = 0,26

Después se calcula el ajuste de complejidad técnica (ACT):

$$ACT = 0,65 + C \times \sum Características$$

C debería calibrarse según la importancia que se quiera dar a las características de ajuste. El valor medio es 0,005

Finalmente PF Mark II = PNA x ACT

PUNTOS DE OBJETO

Otra variación al método inicial de puntos de función es el método de Puntos de Objeto.

Es un método para medir el tamaño funcional a partir de esquemas conceptuales orientados a objeto. Se basa en el método estándar Análisis de Puntos de Función (IFPUG) para el paradigma de orientación a objetos. Se utiliza toda la información disponible durante la fase modelado conceptual. Para ejemplificar el proceso se trabajará sobre el sistema de la Ilustración 3. El proceso es el siguiente:

- Se miden las clases y sus relaciones.

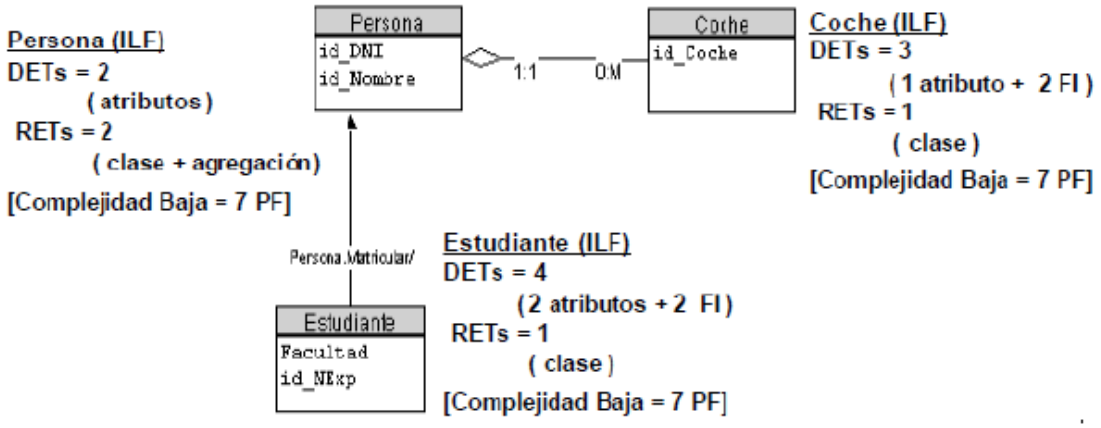

Ilustración 3. Ejemplo para puntos de objeto

- Además de las clases se miden de manera similar la vistas legadas (métodos de una clase que usan atributos de otras clases) y los servicios (similares a las funciones transaccionales, entradas, salidas, consultas). Se rellena la tabla de elementos y su complejidad para contra los PFNA (ver Tabla 8).

Nombre del Proyecto:												
Tipo de Conteo:												
Componente	Complejidad Funcional del Componente											Total
	Baja				Media				Alta			
Clase (ILF)		x 7 =	0		x10 =	0			x 15 =	0		0
Vista legada (EIF)		x 5 =	0		x 7 =	0			x 10 =	0		0
Servicio (EI)		x 3 =	0		x 4 =	0			x 6 =	0		0
					Tamaño Funcional (sin ajustar):							0

Tabla 8. Tabla de cálculo para los puntos de objeto

- Por último se añade complejidad según el modelo de navegación web en el que los nodos son páginas y las arcos vínculos entre las páginas, teniendo en cuenta: número de nodos, número de vínculos, conjunto de atributos y servicios visibles, etc.