Informática de Software

¿Merece la pena usar los puntos de función?

Javier Dolado*, Luis Fernández Sanz**
*Universidad del País Vasco, **Universidad Europea de Madrid

<dolado@si.ehu.es>
<lufern@dpris.esi.uem.es>

Este trabajo se ha desarrollado gracias al apoyo de los proyectos CICYT TIC98 1179-E y UPV EHU 141.226-EA083/98. Asimismo, los autores desean expresar su agradecimiento a María José Rueda por la revisión ortotipográfica del presente texto.

Resumen: *la medición de una aplicación mediante los puntos de función (en adelante, PF) es una de las actividades que más auge está adquiriendo dentro de la ingeniería del software. Sin embargo, la introducción de esta técnica (como la de cualquier otra) se debería hacer con las suficientes garantías de que cumple determinadas propiedades que la dotan de validez, tanto en su estructura teórica como en su aplicación práctica. En este artículo se examinan estas dos cuestiones con respecto al empleo de los puntos de función en la ingeniería del software.*

1. Los puntos de función en la gestión del software

El hecho de cuantificar el tamaño previsto para una aplicación en las primeras fases del ciclo de vida es una preocupación recurrente para todos los jefes de proyecto, puesto que la previsión de esfuerzo suele apoyarse en una estimación previa del tamaño. El uso del número de líneas de código (**LOC**: *lines of code*) como unidad de medida implica que hay que estimar al comienzo del proyecto dicho número, normalmente de forma rudimentaria y poco precisa. Así, los jefes de proyecto tienen que realizar dos estimaciones, una para las LOC y, posteriormente, otra para el esfuerzo. La estimación de las LOC no es una tarea sencilla ya que se trata de una medida del producto final del proyecto que no es fácilmente perceptible hasta que el proyecto tiene un determinado grado de compleción. Aunque existen algunos métodos de estimación de las LOC que han logrado algunos éxitos, no son objeto del presente trabajo.

Una alternativa que se ha explorado es la propuesta de otras medidas del tamaño de la aplicación al comienzo del proyecto. Los PF se han utilizado como alternativa a las LOC y se han convertido en la medida más empleada en la fase de análisis dentro del ciclo de vida de la aplicación. Tanto es así que, recientemente, se ha publicado en una de las revistas de divulgación científica más conocidas un artículo en el que se promociona el uso de **PF** para medir el tamaño del software [Jones, 1998]. Dada la gran difusión que está alcanzando esta técnica, conviene analizar los conceptos en los que se apoya. A continuación vamos a describir cómo aparecieron los PF y de qué modo se mide con ellos.

2. La propuesta inicial: los PF de Albrecht de 1979

La propuesta inicial de los PF fue presentada en un congreso por A. J. Albrecht en 1979 [Albrecht, 1979], si bien documentos internos de IBM parecen indicar que su concepción se remonta, al menos, a 1975[1]. En 1981, para lograr una mayor difusión, su ponencia de 1979 se publicó en un volumen colectivo del IEEE[2] [Jones, 1981]. Según Albrecht, los PF se basan en un estudio de 22 proyectos (16 en COBOL, 4 en PL/I y 2 en DMS/VS) completados entre 1974 y 1979. En esta primera propuesta, se adopta ya el esquema básico de determinación de los PF:

1. Se calculan primero los puntos de función no ajustados (en adelante, PFNA) contando los elementos externos de la aplicación.
2. A continuación se aplica un ajuste de complejidad (en adelante, AC) según el grado de influencia de varios factores.

En esta versión de 1979, los PFNA se obtienen mediante la cuenta de cuatro tipos de elementos: entradas, salidas, consultas (pares de diálogo entrada/respuesta) y ficheros maestro (ficheros o agrupaciones lógicas de datos). Lo normal es que elementos se cuenten a partir de la especificación de la aplicación. Sin embargo, no todos los tipos contribuyen al valor final por igual; para cada uno existe un peso que trata de reflejar su valor funcional para el usuario (**Tabla 1**). Albrecht indica que determinó dichos pesos "mediante debate y prueba" [Albrecht, 1979].

Variables significativas	**Pesos**	**Total sin ajuste**
Número de entradas	∞ 4	_____
Número de salidas	∞ 5	_____
Número de consultas	∞ 4	_____
Número de ficheros maestros	∞ 10	_____
Puntos de Función No Ajustados PFNA =		_____
Ajuste de Complejidad (± 25%)		
N = Factores AC = 0,75 + 0,01 ∞ N =		_____
Puntos de función PF = PFNA ∞ AC =		_____

Tabla 1: Esquema para calcular los PF de 1979

El AC sobre los PFNA se realiza mediante la valoración subjetiva del grado de influencia que tienen en la aplicación diez factores técnicos. La opinión puede oscilar desde el 0 (sin influencia) hasta el 5 (influencia esencial).

3. La versión clásica de más uso: los PF de Albrecht de 1983

Albrecht publicó un artículo en 1983 [Albrecht y Gaffney, 1983] en el que modifica su fórmula de cálculo de los PF. En cuanto a la identificación de elementos, el número de

ficheros se desdobla en dos: número de ficheros lógicos internos y número de ficheros compartidos o pasados a través de la interfaz externa. Por lo tanto, las variables de los PF son ahora las entradas, las salidas, las consultas, los ficheros lógicos internos y los ficheros lógicos externos. Para comprender mejor el modo en que se identifican estos elementos, en la **figura 1** se muestra un esquema sobre cómo se ve una aplicación (aplicación *A*) desde el punto de vista de los PF. La aplicación *A* se comunica con una o varias aplicaciones *B* a través de transacciones e, igualmente, puede compartir determinados ficheros. La aplicación *A* se desarrolla bajo unas determinadas características del entorno (características generales de la aplicación) y se comunicará con los usuarios finales mediante un conjunto de transacciones, que pueden ser entradas, salidas o consultas.

Después de haber identificado los diferentes tipos de elementos, la versión de 1983 requiere asignar a cada uno un valor de "complejidad", que puede ser sencilla, media o compleja.

En el artículo de 1983 simplemente se daban algunas guías para clasificar cada elemento. Así, una salida se considera sencilla si tiene "una o dos columnas" y hay "transformaciones sencillas de datos" [Albrecht y Gaffney, 1983].

Actualmente, tras muchos años de trabajo de especificación, estas guías han llegado a ser algo más objetivas después de elaborar tablas de clasificación [IFPUG, 1994]. Por ejemplo, la **Tabla 2** permite clasificar una salida como sencilla, media o compleja según el número de ficheros y el número de campos de datos a los que hace referencia. En cualquier caso, se asume que la complejidad de cada elemento se mide en una escala ordinal simple de tres valores: sencilla, media o compleja.

Salidas	1-5 ítems de datos referenciados	6-19 ítems de datos referenciados	20 o más ítems de datos referenciados
0 o 1 fichero referenciado	Sencilla	Sencilla	Media
2 o 3 ficheros referenciados	Sencilla	Media	Compleja
4 o más ficheros referenciados	Media	Compleja	Compleja

Tabla 2: Guía de [IFPUG, 1994] para clasificar la complejidad de las salidas de una aplicación

Una vez clasificados los elementos, a cada uno de ellos le corresponde un "factor de ponderación" según su tipo y complejidad (ver **Tabla 3**). No obstante, en ningún sitio (ni en [Albrecht y Gaffney, 1983] ni en otros trabajos) se explican o justifican con criterios objetivos los valores asignados a estos pesos.

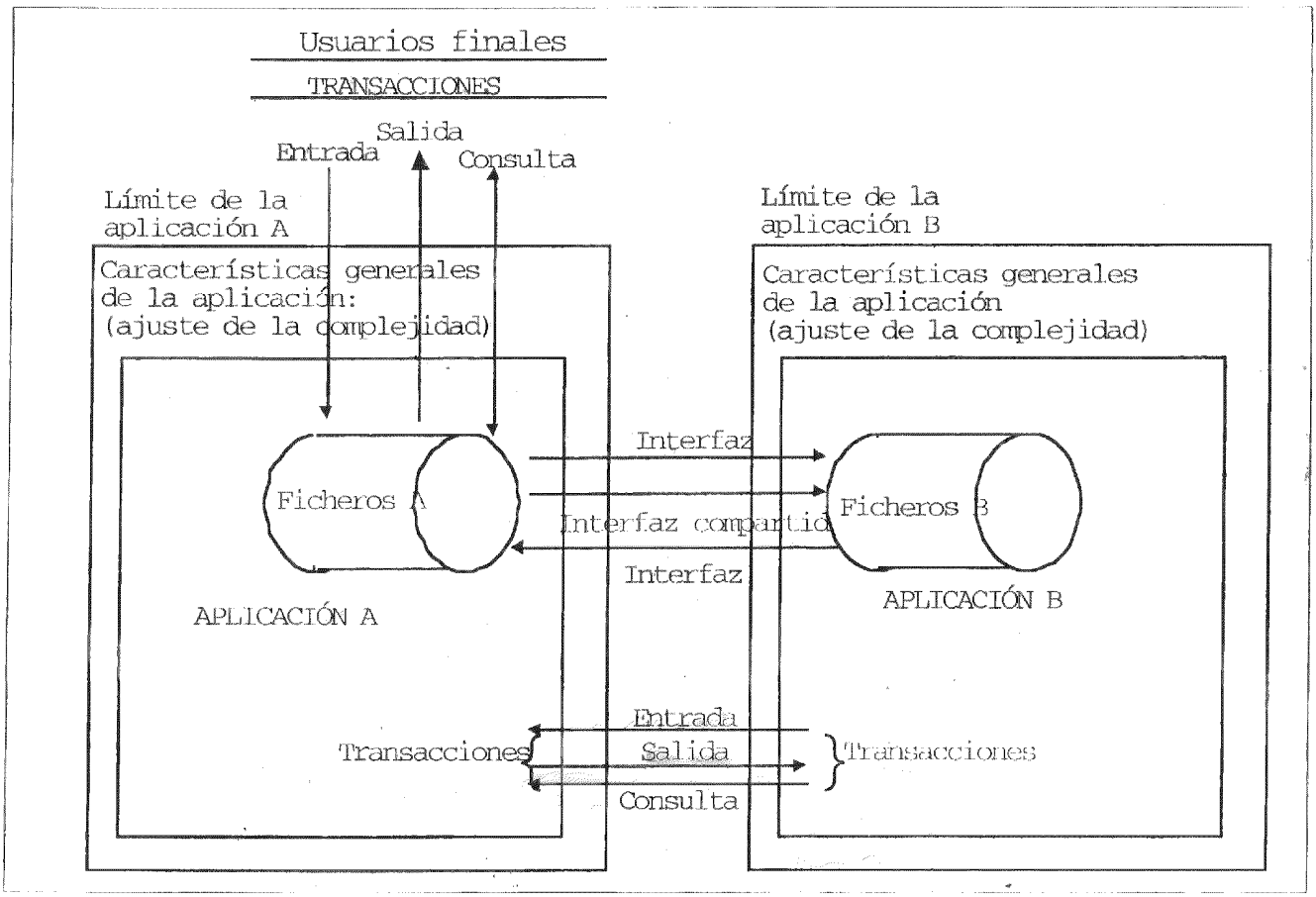

Figura 1: Relaciones entre usuarios, aplicaciones y funciones

	Factor de Ponderación o Complejidad		
Tipo de componente	Simple o baja	Medio	Complejo o alta
Entradas	3	4	5
Salidas	4	5	7
Consultas	3	4	6
Interfaces externas	5	7	10
Ficheros internos	7	10	15

Tabla 3: Pesos para los diferentes componentes

El cálculo de los PFNA es la suma ponderada de todos los elementos (según la **Tabla 2**, tenemos 15 posibilidades) y viene dada por la fórmula:

$$PFNA = \sum_{i=1}^{15} \left(n^{o}\, elementos\, de\, tipo\, i\right) \times \left(peso_i\right)$$

Por último, la cuenta final de los PF supone multiplicar los PFNA por el ajuste de complejidad técnica (ACT). El ACT se calcula puntuando cada factor de la **figura 4** en una escala con los siguientes valores: 0, 1, 2, 3, 4 o 5. El cero significa que el factor es irrelevante para la aplicación y el cinco, que es esencial. Entonces el ACT es:

$$ACT = 0{,}65 + 0{,}01 \sum_{i=1}^{14} F_i$$

Como puede verse, se amplía el número de factores de ajuste de los 10 de la versión de 1979 a los 14 de esta versión de 1983 (**Tabla 4**). La influencia máxima de cada factor sigue siendo del 5%, por lo que la variación que supone el ACT sobre los PFNA puede ser del ± 35%.

Factores que contribuyen a la complejidad de procesamiento (ACT)	
F1: fiabilidad del *back-up* y recuperación	F2: comunicaciones de datos
F3: funciones distribuidas	F4: rendimiento
F5: configuración muy cargada	F6: entrada de datos *on-line*
F7: facilidad de operación	F8: actualización on-line
F9: interfaz compleja	F10: procesamiento complejo
F11: reusabilidad	F12: facilidad de instalación
F13: localización múltiple	F14: facilidad de cambio

Tabla 4: Factores que influyen en el ACT

3. Evolución de los PF: variantes y mejoras

En 1985, la consultora SPR (*Software Productivity Research*) propuso una variante de los PF, la cual contenía las cinco variables de la versión de los PF de 1983, pero sin catalogar cada elemento en sencillo, medio o complejo. En el ACT se reducían los catorce factores a sólo dos (complejidad del problema y complejidad de los datos)[3], evaluados de uno a cinco. No obstante, el ACT puede suponer una variación del ± 40% (frente al 35% de la versión de 1983).

En 1986, SPR lanzó su variante de los PF para software de tiempo real y de sistemas, variante denominada puntos de características (*feature points*). Lamentablemente, en 1991, la técnica estaba "aún en fase experimental y todavía en pruebas" [Jones, 1991].

Más significativa fue la creación en 1986 por parte de una serie de empresas (muchas de ellas, clientes de IBM) de una asociación llamada IFPUG[4] (*International Function Point User Group*) para promover el uso y el intercambio de información sobre los PF. Además de organizar conferencias internacionales, el IFPUG publica documentos con los que se trata de estandarizar las reglas de cálculo de los PF y la terminología manejada. Actualmente se trabaja con la versión 4.0 del manual de cálculo, publicada en 1994 [IFPUG, 1994].

En 1988, C. Symons hizo una dura crítica a los PF de Albrecht [Symons, 1988]. Señaló la subjetividad en la cuenta de elementos, la arbitrariedad de los pesos asignados, la inconsistencia de cálculo de los PF al tratar sistemas divididos en susbsistemas, la ausencia de una cuantificación de la complejidad interna y la necesidad de revisar las 14 características que componen el ACT. Su solución consistió en aportar una nueva variante denominada PF de tipo II (*MarkII Function Points*). En ella se trabaja con el concepto de transacción lógica: una combinación de entrada, proceso y salida, disparada por un acontecimiento de interés para el usuario o por una necesidad de recuperar información. Para cada transacción hay que contar el número de elementos de entrada (*EE*) y de salida (*ES*) que implica, así como las entidades de datos (*ED*) tratadas o referenciadas (**Tabla 5**). A diferencia de Albrecht, Symons indicó que la ponderación (los pesos P_1, P_2 y P_3) de cada variable se debe lograr por calibración en el entorno donde se va a aplicar si bien, en su libro de 1991 [Symons, 1991] ofrece una media de pesos obtenida en diversos proyectos: $P_1 = 0{,}58$, $P_2 = 1{,}66$ y $P_3 = 0{,}26$.

Para cada transacción Ti	$PFNA_i = P_1 \cdot EE_i + P_2 \cdot ES_i + P_3 \cdot ED_i$
PFNA total	$PFNA_T = \sum_i PFNA_i$
ACT: Symons indica que C debe calcularse por calibración (se recomienda 0,005)	$ACT = 0{,}65 + C \cdot \sum factores$
	$PF\ MKII = PFNA \cdot ACT$

Tabla 5: Esquema de cálculo de los PF MarkII

4. Críticas y problemas de los PF

Desde la propuesta inicial de Albrecht, las distintas versiones de los PF han intentado paliar tanto los problemas inherentes al cálculo como los relacionados con las capacidades predictivas de los mismos. Es curioso cómo los propios proponentes de los PF han ido planteando versiones y modificando matices en la técnica, a medida que se iban haciendo conscientes de algunos de sus problemas. También hemos comentado el caso de uno de los autores más críticos con los PF de Albrecht, Symons [Symons, 1988], quien, sorprendentemente, formuló un nuevo método que hereda muchos de los defectos de formulación atribuibles a los PF originales.

La razón por la cual no se ha mejorado la validez de los PF con las distintas propuestas radica en que en ningún caso se han seguido los postulados básicos de la teoría de la medición. Es sólo a partir del proyecto METKIT y de los trabajos de Fenton y sus colaboradores cuando se fundamentan las críticas hacia los PF [Fenton, 1997]. Estos autores demostraron con claridad que su proceso de construcción es matemáticamente incorrecto, hasta tal punto que, precisamente, una de sus principales referencias [Kitchenham *et al.*, 1995] apareció cuando se estaba en proceso de estandarización de la metodología de PF. En dicha referencia trataban de advertir a la comunidad de ingeniería del software sobre los peligros de estandarizar medidas conceptualmente incorrectas. En estos momentos los PF no se han establecido en un estándar, a pesar de la idea que uno se puede formar tras la lectura de artículos como [Jones, 1998]. Lo que sí se está haciendo es estandarizar el proceso de definición de medidas funcionales, de lo que se encarga una parte del estándar ISO/IEC 14143-1 sobre medidas de tamaño funcional [ISO, 1998]. A continuación se describen las deficiencias de definición teórica y de aplicación práctica de las que adolecen los PF.

4.1. Validación teórica, o por qué los PF no miden bien

Para poder medir y operar con los valores de una variable, ésta se debe haber definido de acuerdo con los postulados de la teoría de la medición. Brevemente indicaremos que una variable se puede medir en una escala nominal, ordinal, intervalo, ratio o absoluta. La escala nominal se utiliza con propósitos de identificación (colores, por ejemplo), mientras que en la escala absoluta podemos contar los objetos y establecer un origen o cero absoluto para la cuenta. Lo ideal es que una variable esté medida en una escala absoluta para poder sumar, restar, multiplicar y dividir sus valores. Por el contrario, en una escala nominal sólo cabe diferenciar los elementos; en una escala ordinal únicamente podemos ordenarlos y en una escala intervalo solamente podemos ver la diferencia en el orden establecido. Evidentemente, no está permitido operar con valores de diferentes escalas (por ejemplo, no se puede sumar el color rojo de un objeto con el número de objetos existente).

Sin embargo, ya se ha visto anteriormente en la Sección 2 que en el cálculo de los PF utilizamos varios tipos de escala. Como se ha descrito en [Abran y Robillard, 1996] se puede apreciar que el uso de las escalas nominal, ordinal, ratio y absoluta es inconsistente en el cálculo de los PF, el cual constituye un popurrí de escalas que no es admisible desde el punto de vista teórico. Estos autores también indican que los pesos de los componentes no añaden mucha información al número final, aunque se pueden utilizar sin otros datos históricos en la predicción del esfuerzo.

Por su parte, Kitchenham, Pfleeger y Fenton [Kitchenham *et al.*, 1995] son más críticos con respecto a la validez de los PF e, incluso, proponen abandonar la técnica, dado que no se ajusta a los principios básicos de la teoría de la medida. Sin embargo, la aplicación de la técnica parece extenderse continuamente debido, sobre todo, a la gran base de usuarios de la misma. Esta amplia utilización parece transmitir una cierta confianza sobre su validez práctica a los posibles usuarios, aunque, como veremos en el siguiente apartado, la técnica también falla en este aspecto.

4.2. Validación práctica de los PF, o por qué la medida quizás no sirva para mucho

Es probable que el argumento más importante para usar los PF es el de obtener predicciones del esfuerzo de desarrollo de una aplicación. Por otra parte, también se propone como una medida de tamaño funcional para la valoración económica de las aplicaciones, independientemente de la tecnología.

En cuanto a la predicción del esfuerzo, debemos recordar que una técnica es útil en la medida en que es capaz de predecir ese esfuerzo con fiabilidad. Originalmente, y en palabras textuales de A. Albrecht "...se ha descubierto que los PF son una medida relativa efectiva del valor funcional entregado al usuario" [Albrecht, 1983]. Ahora bien, en este trabajo y en casi todos los posteriores, la bondad de tal efectividad se ha evaluado tanto con el coeficiente R^2 como con el coeficiente de correlación. El R^2 es un criterio de valoración de los modelos de regresión que representa el porcentaje de la varianza justificado por la variable independiente. Se puede interpretar como el cuadrado del coeficiente de correlación de Pearson entre las variables dependiente e independiente, o también como el cuadrado del coeficiente de correlación entre los valores reales de una variable y sus estimaciones. Si todas las observaciones están en la línea de regresión, el valor de R^2 es 1, y si no hay relación lineal entre las variables dependiente e independiente, el valor de R^2 es 0. El coeficiente R^2 es una medida de la relación *lineal* entre dos variables. Los valores que se han obtenido para el coeficiente R^2 en los diferentes estudios publicados sobre los PF varían desde 0,44 hasta 0,87. En uno de los últimos estudios [Jeffery and Stathis, 1996] el coeficiente R^2 varía desde 0,559 hasta 0,662.

Ante estos valores, algunos autores han reafirmado la validez de la técnica. Creemos, sin embargo, que es una conclusión que no se corresponde con la realidad por varios motivos. Tanto el R^2 como el coeficiente de correlación no son las medidas más adecuadas para evaluar la predicción de un modelo; en el mejor de los casos se trata de medidas del ajuste de la ecuación a los datos, no de la capacidad predictiva del modelo. Desde este punto de vista, las variables más convenientes son PRED(0,25), nivel de predicción al 25%, y MMRE, magnitud media del error relativo, definidas en [Conte *et al.*, 1986]. PRED(0,25) se define como el cociente del número de casos en los que las estimaciones están dentro del límite del 25% de los valores reales entre el número total de casos. Por ejemplo, PRED(0,25) = 0,9 quiere decir que el 90% de los casos tiene estimaciones dentro del 25% de sus valores.

El MMRE se define como $MMRE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i - \hat{e}_i}{e}\right|$,

donde e es el valor real de la variable, $\hat{e}$ es su valor estimado y n es el número de proyectos. Así, si el MMRE es pequeño, entonces tenemos un buen conjunto de predicciones. Se

considera que un modelo es aceptable si PRED(0,25) > 0,75 y MMRE < 0,25. Utilizando estos dos criterios la supuesta capacidad de predicción de los PF disminuye notablemente [Dolado y Fernández, 1998]. En algún caso incluso se ha descubierto que no existía correlación entre los PF y esfuerzo de desarrollo, debido principalmente a las diferencias de productividad, que no son compensadas por el entorno de desarrollo [Dolado, 1997].

Además, al manejar los PF, la predicción de esfuerzo tropieza con la ausencia de modelos generales, es decir, no hay una ecuación para cualquier entorno en la que introduciendo el valor de los PF y otros parámetros, obtengamos un valor de esfuerzo o de plazo de tiempo (excepto en el caso de los PF MarkII [Symons, 1991]). Por lo tanto, sería necesario construir un modelo para cada entorno; ni siquiera es posible calibrar uno general, ya que el esfuerzo requerido es equivalente al de construir un modelo nuevo. Esto se puede apreciar incluso en [Albrecht y Gaffney, 1983], donde aparecen ecuaciones diferentes para un entorno de COBOL y para otro de PL/I. Por el contrario, si se opta por convertir los PF a LOC mediante tablas de equivalencia [Jones, 1991] y se aplica un modelo de estimación general (por ejemplo, COCOMO [Boehm, 1981]), aparte de otras dificultades, surge el inconveniente de que existe un doble ajuste de complejidad: por una parte, el ACT de los PF y, por otra, los inductores de coste de COCOMO. A esto hay que añadir que, los modelos generales de estimación de costes están plagados de múltiples problemas: poca precisión, dificultades de operación debidas a una mala definición de sus elementos, deficiencias en su modo de construcción y muy poca flexibilidad para adaptarse a otros entornos, a pesar de la calibración y el uso de factores de coste [Fernández, 1997].

Lamentablemente, los problemas de la técnica de los PF no terminan aquí. Hemos de mencionar dificultades que aparecen, en especial, cuando se utilizan como criterio de valoración económica, pero también cuando sirven de base para la predicción [Fenton y Pfleeger, 1997]:

a) La subjetividad en la aplicación del ACT.
b) La doble cuenta de la complejidad según se ha explicado en el apartado anterior.
c) Los valores anti-intuitivos que aparecen cuando se asigna el valor medio a los factores del ACT.
d) La dificultad para su utilización cuando no existe una especificación completa.
e) Las diferencias entre el valor de PF obtenido sobre la especificación original de un sistema y el obtenido cuando se aplica sobre el sistema ya terminado.
f) Los PF sólo están destinados a aplicaciones de gestión ya que la efectividad de las variantes para otros tipos de software no se ha justificado suficientemente
g) Diferentes personas obtienen valores distintos al contar los elementos para calcular los PF, a pesar de las guías de IFPUG.
h) La correlación (colinealidad) entre los elementos constitutivos de los PF (entradas, salidas, etc.) cuestiona su utilidad; también se observa una escasa aportación del ACT a la predicción, puesto que los PFNA proporcionan valores de predicción similares a los obtenidos con los PF finales [Jeffery and Stathis, 1996].

En definitiva, son demasiados problemas como para aceptar, sin más, la aplicación de la técnica de los PF. Por lo tanto, debemos ser cuidadosos al interpretar los valores sobre PF que aparecenen artículos como [Jones, 1998], puesto que esos datos pueden proporcionar una falsa sensación de seguridad sobre la capacidad de la técnica, sobre todo cuando, desde el punto de vista científico, exigimos en primer lugar una correcta definición del concepto para más tarde utilizarlo en la práctica. Desgraciadamente, esta cuestión se elude en un artículo de tan amplia repercusión como el ya mencionado [Jones, 1998].

5. ¿Hay otras posibilidades?

Una vez que hemos analizado los problemas que plantean los PF, se nos plantea la cuestión de cuál o cuáles pueden ser las alternativas válidas. A cualquier nueva propuesta se le debe pedir, al menos, las mismas propiedades matemáticas que las exigidas a los PF. Para ello, debemos emplear en todo momento una escala ratio, es decir, recurrir siempre a la cuenta de entidades o de elementos, sin utilizar tablas de valoración de los mismos.

No son muchas las alternativas que se han publicado; incluso, alguna de ellas cae en los mismos errores que los cometidos en los PF, como es el caso de la de los Puntos Objeto (PO) [Boehm, 1995]. Se basan en definir unos objetos utilizados en un entorno determinado que, posteriormente, se pueden asociar a los componentes de los PF. Su principal utilidad consiste en contar con un "repositorio de objetos" que representan a los elementos del sistema de información. De este conjunto de objetos, para el cálculo de los PO hay que estimar el número de pantallas, de informes y de componentes 3GL (elementos procedimentales). Como en los PF, los elementos se clasifican en sencillos, medios o difíciles, y se les asigna un peso de complejidad.

En [Dolado, 1998] se comentan otras alternativas posibles como el manejo de modelos generales de estimación de tamaño, la cuenta de diferentes elementos en la orientación a objetos, la medición sobre metamodelos, la medición basada en el recuento de diversos tipos de componentes, las medidas sobre los diagramas de flujo de datos, etc. Se puede afirmar, en definitiva, que es posible contar otros objetos identificables en una especificación de software con igual o mejor capacidad de predicción y medición que los PF. Por eso, recomendamos la elaboración específica de ecuaciones en cada entorno en el que se desee establecer una relación entre una medida de tamaño de la especificación del sistema y el esfuerzo preciso para su desarrollo.

6. Conclusión

Supongamos que medimos habitualmente la misma clase de tela con la misma regla. Aunque el centímetro que usemos como unidad no coincida con el que se emplee oficialmente, siempre obtendremos una información interesante para comparar las diferentes telas medidas. Así, nos resultará posible tomar decisiones basándonos en esa información (por ejemplo, cuántas piezas de tela necesitamos para hacer un traje). El problema aparece cuando queremos comunicar a otra persona cuánto mide nuestra tela en "centímetros de

verdad". Si nuestro centímetro no coincide con el de la otra persona, es imposible una correcta comparación. Tampoco es posible saber cuántos metros de tela tenemos si las telas están arrugadas y no las estiramos para medirlas. En términos de PF, estas analogías equivalen, en primer lugar, al hecho de contar o no contar determinados elementos y, en segundo, a la constatación de que dos analistas pueden valorar la misma especificación de distinta manera.

Si el sastre mide las telas a ojo, aunque intente todos los días medir igual mancra, es muy difícil que la medida sea siempre exactamente la misma. Por lo tanto, lo que mide un día puede diferir del valor obtenido al día siguiente para la misma pieza de tela. En relación con los PF, esto significa que un analista puede fácilmente pasar por alto alguno de los elementos necesarios para el cálculo de los PF.

Utilizar los PF no resuelve los problemas de fondo de los proyectos. Lo que se debe hacer es establecer un plan de medición que defina las características de las especificaciones. Después, se ha de analizar en cada entorno y en cada proyecto si dichas propiedades son o no indicadores del esfuerzo de desarrollo o de cualquier otro atributo que se desee evaluar o predecir [Fernández, 1998].

6. Referencias

A. Abran y P.N. Robillard; Function points analysis: an empirical study of its measurement processes, *IEEE Trans. on Software Engineering*, vol. 22, nº 12, pp. 895-909, 1996.

A.J. Albrecht; Measuring application development productivity, *Proceedings of the Joint SHARE, GUIDE and IBM Application Development Symposium*, octubre 1979, pp. 83-92.

A.J. Albrecht y J.E.Gaffney; Software function, source lines of code, and development effort prediction: a software science validation, *IEEE Transactions on Software Engineering*, vol. 9, nº 6, noviembre 1983, pp. 639-647.

B. Boehm *et al.*; Cost models for future software life cycle processes: Cocomo 2.0, *Annals of Software Engineering*, vol. 1, 1995, pp. 57-94.

J.J. Dolado; A study of the relationships among Albrecht and Mark II function points, lines of code 4GL and effort, *Journal of Systems and Software*, vol. 37, pp. 161-173, 1997.

J.J. Dolado; Medición de especificaciones de software, en *I Cuaderno de Calidad*, Novática, enero/febrero, 1999.

J.J. Dolado y L. Fernández; Genetic programming, neural networks and linear regression in software project estimation, *INSPIRE III*, pp 157-171, 1998.

N. Fenton y S.L. Pfleeger; *Software Metrics. A rigurous and practical approach*, PWS Publishing, 1997.

L. Fernández; *Medición y predicción de atributos de diseño en una metodología formalizada*, tesis doctoral, Universidad del País Vasco, 1997.

L. Fernández; Una revisión breve de la medición del software, en *I Cuaderno de Calidad*, Novática, enero/febrero, 1999.

International Function Point User Group; *Function point counting rules 4.0*, IFPUG, enero, 1994.

ISO; *ISO/IEC 14143-1:1998. Information technology: Software measurement. Functional size measurement. Part 1: Definition of concepts*, ISO, 1998.

R. Jeffery y J. Stathis; *Function Point sizing: structure, validity and applicability, empirical software engineering*, vol. 1, pp. 11-30, 1996.

C. Jones; *Programming productivity-Issues for the Eigthies*, IEEE Computer Society, 1981, pp. 35-44.

C. Jones; *Applied software measurement-Assuring productivity and quality*, McGraw-Hill, 1991.

C. Jones; "Sizing up software", *Scientific American*, diciembre 1998 (traducido en "Evaluación del tamaño del software", *Investigación y Ciencia*, febrero 1999).

B.A. Kitchenham, , S.L. Pfleeger y N. Fenton; Towards a Framework for Software Measurement Validation, *IEEE Transactions on Software Engineering*, vol. 21, nº 12, pp. 929-944, 1995.

C.R. Symons; Function point analysis: difficulties and improvements, *IEEE Transactions on Software Engineering*, vol. 14, nº 1, 1988, pp 2-11.

C.R. Symons; *Software sizing and estimating. MkII function point analysis*, Wiley and Sons, 1991.

Notas

[1] Según se indica en [Jones, 1991], existen documentos de 1975 sobre los PF, como: IBM Corporation, *DP services size and complexity factor estimator*, DP Services Technical Council, 1975.

[2] *Institute of Electrical and Electronics Engineers.*

[3] Se incluye una tercera característica de complejidad de código cuando se aplica esta versión al software ya construido (*backfire FP*) con el propósito de realizar estudios históricos de productividad.

[4] Para más información, se puede acceder al servidor de IFPUG en *http://ifpug.org.*