Medición de Especificaciones de Software *

José Javier Dolado Cosín
Facultad de Informática, Universidad del País Vasco

dolado@acm.org

*Artículo elaborado bajo la subvención de la acción especial CICYT TIC98-1179-E

Resumen: *una de las tareas críticas al comienzo de un proyecto informático es la medición o estimación del tamaño de la aplicación, puesto que muchas decisiones posteriores del jefe del proyecto se basan en esa estimación inicial. En este trabajo se describen las principales medidas existentes de las especificaciones de software.*

1. La Estimación del Tamaño en la Gestión de Proyectos

La variable **tamaño de una aplicación** es un elemento clave en los actuales modelos de estimación del coste del desarrollo de software, siendo éste último el factor más importante en el control del proyecto. La incertidumbre en la estimación del tamaño y del esfuerzo tiene un rango muy elevado en las primeras fases del proyecto. Ese rango se va estrechando a lo largo del proyecto de tal forma que quedan completamente definidas dichas variables al final del proyecto. Idealmente, lo que desearíamos es tener una **medida** en vez de una **estimación**, ya que en la primera están reducidas al mínimo las imprecisiones.

Puesto que no es objeto del presente trabajo el incidir en la teoría de la medición, sucintamente diremos que los conceptos de "medición" y "estimación" están ligados mediante el concepto de **sistema de predicción**, en el que una variable independiente (o más) que ha sido medida permite predecir el valor de otra que será una estimación de los valores **desconocidos** de la variable dependiente.

En algunos casos la estimación puede no utilizar ninguna otra variable de soporte. Por ejemplo, supongamos que queremos estimar el número de líneas de código -**LOC**- de una aplicación en las primeras fases del proyecto. Un ejemplo muy sencillo de modelo de estimación (utilizado en modelos como el de Putnam [Putnam, 1992]) es el que viene dado por la siguiente ecuación ELOC = (X1 + 4M + X2)/6, donde X2 el valor máximo previsto, X1 es el menor valor previsto y M es el valor más probable. El valor estimado es ELOC. La desviación típica viene dada por (X2-X1)/6. Alternativamente podemos utilizar otros modelos algorítmicos para estimar las LOC partiendo de alguna de las medidas que se presentan en los apartados siguientes.

En la fase de planificación de un proyecto la medición contiene, a veces, una parte de estimación, puesto que algunas valoraciones consisten en estimaciones del número de objetos o de su tamaño. Dado que es imposible saber con total precisión las LOC de una aplicación al comienzo del desarrollo, son necesarias otras medidas que se puedan construir sobre los documentos existentes en ese momento.

A continuación se exponen los métodos más conocidos de medición y/o estimación de las especificaciones. Más que inclinarnos por una en particular, la idea es presentar los aspectos más importantes de cada una de ellas. La posibilidad de medición de una especificación depende de la cantidad de características identificables en la misma. En la medida en que las actividades realizadas en la fase de especificación tiendan a construir elementos claramente definidos para actividades posteriores se conseguirá realizar una medición más que una estimación.

Los elementos identificables pueden ser objetos, entidades, componentes o cualquier otra cosa. Su utilidad en actividades ulteriores del proyecto es la que suele guiar la forma en la que se definen o representan. Cualquiera de estas representaciones se puede utilizar como base para una estimación del esfuerzo. Aunque se habla en términos de **especificaciones**, algunos de los constituyentes de las medidas analizadas en este artículo pueden pertenecer a la fase de diseño.

2. Primera Aproximación: Puntos de Función Albrecht

Los puntos de función (o **puntos función, PF**) fueron la primera propuesta de medición alternativa a la estimación de las LOC en la fase de análisis. Su primera versión fue formulada por A. Albrecht en 1979 aunque publicó un trabajo más elaborado en 1983 [Albrecht, 1983]. Sobre ese esquema fun-damental han ido apareciendo tanto versiones alternativas como sucesivas revisiones, siendo la última generalmente admitida la descrita por el **IFPUG** (*International Function Points Users Group*) en su versión 4 [IFPUG, 1994].

La métrica de los puntos de función trata de medir la **cantidad de función** existente en un producto software, independientemente del lenguaje y tan pronto como sea posible en el desarrollo de un proyecto. Dado que existe abundante bibliografía sobre este tema sólo vamos a describir brevemente el esquema de cálculo. Los pasos a seguir son: a) identificación de los componentes; b) clasificación de los mismos; c) cálculo de los puntos de función no ajustados -PFNA-; y d) ajuste de los puntos mediante 14 características generales. Consiguientemente, se deben identificar primero, contándolos o estimándolos, los elementos de una aplicación según las siguientes categorías: entradas, salidas, consultas, interfaces y ficheros lógicos internos. A cada uno de esos elementos debe asignárseles un tipo de complejidad como **baja** (o simple), **media** o **alta** (compleja) **(tabla 1)**. Esta asignación se realiza utilizando otras tablas para las que se deben utilizar variables como número de

elementos de datos, número de claves, etc.

Tipo de componente	Complejidad Baja	Media	Alta
Entrada externa	3	4	5
Salidas externas	4	5	7
Consulta externa	3	4	6
Interfaces externas	5	7	10
Ficheros Internos	7	10	15

Tabla 1. Pesos para los diferentes componentes

Los PFNA son la suma de todos los pesos obtenidos en los componentes. El siguiente paso es valorar, en una escala de 0 a 5, catorce características que describen el entorno en el que se desarrolla la aplicación -CAR-. El factor de ajuste a aplicar a los PFNA viene dado por FA = 0.65 + 0.01 ● CAR. Los puntos de función finales son PF = PFNA ● FA.

Esta estructura de cálculo se ha mantenido más o menos estable para los puntos de función Albrecht. Sin embargo, su aplicación a diferentes entornos distintos del original ha requerido el diseño de nuevos tipos de puntos de función. Las principales variedades aparecidas son los puntos MarkII propuestos por Symons [Symons, 1991], los SPQR, y últimamente los *Full Function Points* (**FFP**) [Desharnais, 1997]. Los FFP tratan de incorporar al cálculo las propiedades específicas de las aplicaciones en tiempo real y de control (todavía no han sido ratificados formalmente por el IFPUG).

Los PF son la medida de las especificaciones más extendida, aunque se ha demostrado que su construcción matemática es incorrecta (por ejemplo, [Kitchenham, 1995]). Diversos estudios mostraron inicialmente cierta correlación estadística con el esfuerzo de desarrollo (por ejemplo, [Albrecht, 1983]), si bien desde el punto de vista predictivo la correlación no es la variable más importante a considerar. Hay que destacar la gran cantidad de publicaciones y de grupos de interés sobre los PF.

Las siguientes medidas son todas, con gran diferencia, menos conocidas, aunque no por ello pueden dejar de ser válidas.

3. Puntos Objeto

Se basan en la definición de unos objetos utilizados un entorno determinado, que posteriormente se pueden asociar a los componentes de los puntos de función [Banker, 1994]. Se utilizan en el modelo de estimación COCOMO 2.0 [Boehm, 1995]. La principal utilidad es la de poder tener un **repositorio de objetos** que representan los datos y objetos que conforman ese sistema de información. Los **Puntos Objeto (PO)** se han propuesto debido a las prácticas utilizadas en la construcción de aplicaciones a partir de componentes, especialmente en entornos de desarrollo integrados -ICASE-, donde se utilizan ayudas a la construcción de interfaces y a la reutilización de los componentes. En un estudio realizado por Banker et al. [Banker, 1994] sobre 19 proyectos se descubrió que los PO proporcionaron resultados comparables a los PF. Los pasos para el cálculo de los PO son parecidos a los seguidos en el cálculo de los PF.

Paso 1. Establecer la **cantidad de objetos**. Puede tratarse de una estimación o de una simple cuantificación del número de pantallas, informes, y objetos 3GL, componentes todos ellos que formarán parte de la aplicación.
Paso 2. Clasificar cada instancia de objeto como de complejidad **simple**, **media** o **compleja** de acuerdo con unas tablas en la que se especifican diferentes criterios de complejidad.
Paso 3. Ponderar el valor de cada casilla de la tabla anterior según el esquema de la **tabla 2**. Estos pesos reflejan el esfuerzo relativo requerido para implementar una instancia de ese nivel de complejidad concreto.
Paso 4. Determinar los PO: consiste en sumar todas las instancias ponderadas de los objetos para obtener un número, que será la cuenta de los PO.
Paso 5. En este paso se tiene en cuenta la reutilización en el desarrollo. Se estima el porcentaje de reutilización del proyecto y se calculan los Nuevos Puntos de Objeto -NOP- como NOP = PO ● (100 - % reutilización) / 100.

Tipo de Objeto	Peso de complejidad Simple	Media	Difícil
Pantalla	1	2	3
Informe	2	5	8
Componente 3GL	-	-	10

Tabla 2

Como se puede observar, el esquema de cálculo es similar al de los PF, incluyendo también en el número obtenido una cierta información sobre el esfuerzo. La principal diferencia con los anteriores es que se incluye expresamente la reutilización, por lo que esta medida está claramente enfocada al cálculo del esfuerzo en el propio esquema de construcción.

4. Método de propósito general de Hakuta et al.

Este método, propuesto recientemente, es más un modelo de estimación que de medición [Hakuta, 1997]. Se trata de un modelo de estimación **universal**, independiente del tipo o características del programa. Los elementos fundamentales del método son: 1) unidades de procesamiento, que son módulos de programa que realizan una función concreta, y que se determinan a partir de la especificación del programa; 2) complejidad del procesamiento, que refleja determinadas posibilidades del diseño del programa; y 3) factores de entorno que afectan al tamaño final del programa.

La idea de estimación de tamaño consiste en utilizar principalmente las unidades de procesamiento como factores básicos de estimación de las LOC. La identificación de las diferentes unidades de procesamiento se realiza sobre la especificación del diseño funcional y de otros documentos descriptivos de la aplicación. Para cada función identificada se debe asignar un tamaño de referencia y unos límites superior e inferior (las fórmulas específicas que hay que aplicar quedan fuera del objeto de este artículo y se pueden consultar en la referencia). Los elementos 2) y 3) contienen parámetros que modificarán el valor estándar de 1), mediante los que se ajusta la estimación inicial.

Este método de estimación de tamaño solamente utiliza la parte estrictamente de medición en la cuantificación y clasificación de las unidades de procesamiento. Los resultados logrados por

sus autores han sido buenos. Cabe destacar que la principal característica, por su generalidad, es que la estimación se puede realizar en cualquier fase del desarrollo.

5. Orientación a Objetos

En la orientación a objetos ha habido diversos intentos de medición de las características de los diseños y de su tamaño. Medidas elementales de tamaño como pueden ser simplemente la cantidad del número de clases, de métodos, profundidad de la herencia y otros elementos del diseño son medidas habitualmente utilizadas en la gestión de proyectos.

Jenson y Bartley [Jenson, 1991] propusieron una descripción abstracta orientada a objetos con el propósito de evitar la estimación de esfuerzo a través de las LOC. Su visión de un modelo OO de una aplicación informática consiste en considerar como componentes fundamentales los objetos, las operaciones sobre esos objetos y las interfaces entre objetos.

Las interfaces son las dependencias de los objetos con otros objetos, algo similar a una relación entre objetos. Estos componentes son abstracciones de sus correspondientes conceptos en el mundo real. Simplemente con la cuantificación de las instancias de estas tres variables, se puede construir un modelo de estimación de esfuerzo.

Moser ha propuesto un modelo de medición basado en metamodelos [Moser, 1995] específicamente orientado a objetos, con el propósito de sustituir a los PF. Su idea consiste en construir un modelo de empresa en el ámbito de la especificación. Para cada objeto considera un **tamaño externo** y un **tamaño interno**, mediante la cuantificación de los elementos que componen cada objeto. Sus resultados en la predicción del esfuerzo han sido ligeramente superiores a los PF.

6. Medición y Estimación basada en Componentes

Los trabajos de J. Verner y G. Tate han servido para modificar la visión estática sobre los tipos de componentes identificables en una especificación [Verner, 1992]. Su idea fundamental es que los tipos de componentes que se pueden identificar en las primeras fases de una aplicación no tienen por qué estar prefijados y pueden depender de la tecnología. A partir de esta idea básica estos autores construyen modelos de estimación de tamaño en LOC para cada tipo de componente. Sus resultados indican que es mejor estimar el tamaño *bottom-up*, por tipo de componente, que globalmente. La medición se realiza por la simple cuantificación de los tipos de componentes y de alguna de las características que los definen (número de tablas, opciones, etc.). Este modelo ha sido validado favorablemente en [Dolado, 1999].

7. Otras medidas

Además de las medidas arriba mencionadas también se puede mencionar la medida *Bang*, propuesta por Tom DeMarco, basada en los diagramas de flujo de datos **-DFD**. Tiene en cuenta la complejidad de los DFD y los tipos de operaciones (funciones) que operan en ellos. Recientemente se ha propuesto una medida, matemáticamente bien construida, que asocia a determinado tipo de DFD una Red de Petri, de tal modo que el valor de la métrica permite comparar correctamente dos DFDs en cuanto a su tamaño [Fernández, 1998]. Su desarrollo formal es correcto, aunque todavía es necesaria una más amplia validación empírica.

8. Conclusiones

Como hemos podido comprobar existen varias posibilidades de medición de las especificaciones. Las más sencillas se basan simplemente en contar determinados elementos que son visibles en la fase de análisis. Otras, más elaboradas, generan un número ponderando elementos más básicos.

La elección de cualquiera de las medidas presentadas debe hacerse teniendo en cuenta el entorno y las experiencias previas con esa medida. Aunque varias de las técnicas se proclaman como independientes de la tecnología, lo cierto es que todas plantean problemas en su utilización práctica.

Algunos de los elementos que aparecen en las especificaciones sólo se pueden ponderar adecuadamente una vez analizadas características que aparecen en el diseño. Incluso en algunos casos es preciso definir con detalle las características de manipulación y tipo de los datos, lo que hace que la medición de la especificación incluya bastante más información que la simple descripción de los requisitos generales.

La estimación del tamaño es un problema todavía abierto. Los métodos o modelos de medición/estimación propuestos sólo pueden ayudar al "ojo clínico" del jefe del proyecto, que es el único que con su experiencia y conocimiento puede ajustar la estimación.

8. Referencias

Albrecht, A.J. y J.E. Gaffney, Software Functions, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, *IEEE Trans. Software Eng.*, vol. 9, no. 6, Noviembre, 1983.

Banker, R.D., R.J. Kauffman, C. Wright y D. Zweig, Automating Output Size and Reuse Metrics in a Repository-Based Computer-Aided Software Engineering (CASE), *IEEE Trans. on Software Engineering*, Vol.20, no. 3, pp. 169-187, March, 1994.

Boehm, B. et al., Cost Models for Future Software Life Cycle Processes: COCOMO 2.0, *Annals of Software Engineering*, Vol. 1, pp. 57-94, 1995.

Desharnais, J-M., D. St-Pierre, M. Maya y A. Abran, Full Function Points: Counting Practices Manual, Rules and Procedures, Université du Québec à Montréal, Montréal, *Technical Report* no. 1997-06, November, 1997.

Dolado, J.J., A Validation of the Component-Based Method for Software Size Estimation, *IEEE Trans. on Software Engineering*, por aparecer en 1999.

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

Fernández, L. y J.J. Dolado, *Measurement and Prediction of Design Attributes in a Formalized Methodology*, enviado para publicación.

Hakuta, M., F. Tone y M. Ohminami, A Software Size Estimation Model and Its Evaluation, *Journal of Systems and Software*, 37:253-263, 1997.

International Function Point Users Group (IFPUG), *Function Point Counting Practices Manual*, Release 4.0, IFPUG, Westerville, Ohio, Jan., 1994.

Jenson, R.L. y J.W. Bartley, Parametric Estimation of Programming Effort: An Object-Oriented Model, *Journal of Systems and Software*, 15:107-114, 1991.

Lorenz, M. y J. Kidd, *Object-Oriented Software Metrics*, Prentice-Hall, 1994.

Moser, S. Metamodels for Object-Oriented Systems, *Software - Concepts and Tools*, , pp. 63-80, 1995.

Putnam, L.H., y W. Myers, *Measures for Excellence. Reliable Software on Time, within Budget*, Prentice-Hall, 1992.

Symons, C.R. *Software Sizing and Estimating. MkII Function Point Analysis*, John Wiley & Sons, 1991.

Verner, J. y G. Tate, "A Software Size Model", *IEEE Trans. on Software Engineering* 18, no. 4 April, pp. 265-278, 1992.