CAPÍTULO 6

MÉTRICAS DE LA REUTILIZACIÓN EN ORIENTACIÓN AL OBJETO

Mª Esperanza Manso Martínez
Dpto. de Informática, U. de Valladolid manso@infor.uva.es
Francisco José García Peñalvo
Dpto. de Informática y Automática, U. de Salamanca fgarcia@gugu.usal.es

RESUMEN

La reutilización del software es uno de los medios que permiten mejorar la productividad y la calidad de los productos. Esta premisa se ve confirmada en numerosas referencias de la bibliografía especializada. Esto supone plantear modelos que permitan medir cuánto se ha reutilizado, cómo se ha reutilizado, y cómo se han modificado los costes o esfuerzo del desarrollo. En este capítulo exponemos conceptos básicos sobre reutilización del software, parte de los cuales son propios de la reutilización en el paradigma orientado a objetos. Además, planteamos diferentes aproximaciones a la reutilización a través de modelos y métricas que permiten medir atributos, tanto del proceso como del producto, relacionados con la reutilización. Aunque algunos de los modelos o métricas que se plantean no se desarrollaron expresamente para la orientación al objeto, pueden aplicarse con igual validez a este paradigma.

1. INTRODUCCIÓN

La reutilización del software se está vendiendo como una de las mejores aproximaciones para atajar la tan afamada crisis del software, como una posible solución para dar respuesta a un mercado que exige productos y servicios cada vez más fiables, baratos y entregados a tiempo [Karlsson, 1995], [Jacobson et al., 1997] y [McClure, 1997]. La reutilización del software no es un planteamiento nuevo – las bibliotecas de funciones son un ejemplo de reutilización de código – pero se ha potenciado con el paradigma de orientación al objeto. Los conceptos de clase, herencia y polimorfismo constituyen soportes básicos de la reutilización.

En diferentes publicaciones se exponen los beneficios de la reutilización que, de una u otra forma, se pueden traducir en mejora de los costes o producción e incremento de la calidad del producto software, pero en ocasiones no está claro a qué se llama reutilización ni cómo se está midiendo la mejora de costes o el incremento de calidad. Es evidente que si desconocemos cuanto hemos reutilizado, mal podremos cuantificar el beneficio que aporta dicha reutilización. Sin embargo, definir que se va a contar como software reutilizado y medir los beneficios que aporta no es trivial.

En un estudio reciente, desarrolladores de software que habían adoptado tecnología orientada a objetos para explotar sus posibilidades de reutilización, manifestaron que el 37% de las 300 compañías allí representadas tenían como objetivo primero la

reutilización de software, el 33% buscaban mejorar la calidad del código y el 29% esperaban reducir el tiempo de desarrollo. Para conseguirlo un tercio de las compañías estaba invirtiendo en orientación al objeto. Un 24% reutilizaba cortando y pegando y sólo un 9% disponía de un programa de reutilización sistemática [Poulin, 1996]. Esto abre interrogantes sobre el control que las compañías tienen sobre el software reutilizable y sobre la consistencia de las métricas de reutilización que usan, además pone de manifiesto la ausencia de reutilización en las primeras fases del ciclo de vida del proyecto.

El propósito de este capítulo es presentar una visión panorámica sobre los beneficios que aporta la reutilización en la Ingeniería del Software, y el papel que en este campo juega la medición. Para ello presentaremos, en el apartado 2º, una clasificación de la reutilización en diferentes tipos y definición de conceptos básicos. En el apartado 3º una introducción a las métricas en reutilización. En los apartados 4º, 5º y 6º presentamos las métricas que son de interés en reutilización: del nivel de reutilización, reusabilidad y económicas respectivamente. En los apartados 7º y 8º se encuentran las conclusiones y las referencias bibliográficas, respectivamente.

2. REUTILIZACIÓN:CONCEPTOS BÁSICOS Y CLASIFICACIÓN

"*La reutilización de software es el uso de elementos software existentes, durante la construcción de nuevos sistemas de software. El enfoque de la reutilización no es sólo el código fuente, sino cualquier producto intermedio generado en el proceso de desarrollo...*" [Freeman, 1983]. Para Karlsson [Karlsson, 1995, p.255] un elemento software reutilizable es "*cualquier componente desarrollada específicamente para ser usada, y que actualmente es utilizada en más de un contexto*". El concepto de reusabilidad de un elemento software esta relacionado con el de reutilización, pero no debe confundirse con éste. La reusabilidad de un elemento software podemos definirla como "*una medida de la facilidad con la que pueden utilizarse conceptos o elementos software en nuevas situaciones*" [Prieto-Díaz et al., 1987].

La reutilización del software se puede clasificar desde diferentes puntos de vista o dimensiones, ortogonales entre sí, que nos permiten conocer facetas de la reutilización susceptibles de medirse, y por tanto, mejorar nuestro conocimiento sobre ella. Se pueden encontrar clasificaciones en [Prieto-Díaz, 1993], [Frakes et al., 1996] y [Poulin, 1996] entre otros. En la Tabla 6.1 se muestra una de las posibles clasificaciones, cuyos conceptos definiremos a continuación.

ámbito de desarrollo

Una dimensión de la reutilización tiene que ver con la procedencia del elemento software a reutilizar: si es *interno (privado)* a la organización (o aplicación) o es *externo (público)* a ella. Para poder realizar esta clasificación las fronteras de la organización o aplicación deben estar bien definidas. Pero estas fronteras, dentro de la orientación al objeto, no son siempre fáciles de determinar, por lo que la reutilización con modificaciones es práctica acostumbrada en este paradigma.

modificación

Cuando los elementos software se reutilizan sin modificar nos referiremos a reutilización de tipo *caja negra o verbatim*, junto con la reutilización sistemática es la que aporta mayores beneficios. Poulin [Poulin, 1996] pone de manifiesto los beneficios de la caja negra frente a la caja blanca, si bien admite reutilización con cierto tipo de modificaciones en el paradigma de orientación al objeto (ver tabla 6.3). En este caso, las modificaciones del comportamiento de un componente se pueden hacer a través de los mecanismos propios del paradigma orientado a objeto (herencia, polimorfismo...) que le van a proporcionar la capacidad para ser reutilizado.

ámbito de dominio

El software que se reutiliza puede proceder del mismo dominio o área de aplicación y la reutilización es *vertical*. "*El objetivo de este tipo de reutilización es derivar modelos genéricos para familias de sistemas. Cuanto más estrecho sea el dominio mayor será el beneficio que proporcione la reutilización*" [Prieto-Díaz, 1993]. Si se reutilizan componentes de diferentes dominios la reutilización es *horizontal*; en este caso los elementos reutilizables son genéricos, independientes del dominio, como por ejemplo: *tipos abstractos de datos, bibliotecas matemáticas, herramientas UNIX...*

nivel de granularidad

El nivel de granularidad de un elemento software reutilizable hace referencia a si es un elemento atómico (grano fino) o esta compuesto por un conjunto de elementos reutilizables (grano grueso). Como ejemplo de grano grueso podemos destacar los mecanos[1] [García et al., 1999], cuya característica más destacable es el soporte simultáneo de diferentes niveles de abstracción

gestión

Una organización gestiona la reutilización de forma *sistemática* si la planifica e incorpora como parte del proceso de desarrollo. La gestión es *casual* o *ad hoc* si la reutilización se produce accidentalmente. Cualquier organización puede practicar esta segunda forma de reutilización, pero los beneficios que aporta raramente se miden.

perspectiva cliente-servidor-sistema, desarrollo con-para

Desde la perspectiva del desarrollo de una aplicación podemos plantearnos desarrollo *con* reutilización (creación de aplicaciones software utilizando elementos ya existentes) o desarrollo *para* reutilización (procesos destinados a la creación de elementos reutilizables). En el desarrollo con y para reutilización juega un papel fundamental el concepto de repositorio, entendiendo por tal "*la base de datos que permite almacenar componentes reutilizables, junto con la documentación y la clasificación asociadas*" [Karlsson, 1995, p.494].

---

[1] Este trabajo ha sido realizado dentro del ámbito del proyecto **MENHIR** (proyecto **CICYT TIC97-0593-C05-05**)

En orientación al objeto es frecuente la perspectiva *cliente-servidor-sistema* [Bieman, 1991], [Frakes et al., 1996], que es similar a la anterior clasificación. La perspectiva del cliente es la de una nueva aplicación (o componente o clase) que reutiliza un componente o una clase (desarrollo con reutilización).

reutilización directa-indirecta

La reutilización de un componente es directa cuando se lleva a cabo sin intermediarios. En otro caso es indirecta.

Ámbito de Desarrollo	**Modificación**	**Ámbito de Dominio**	**Nivel de granularidad**	**Entidad reutilizada**	**Gestión**	**Perspectiva**	**Reutilización**
Interno (privado)	Caja Blanca	Vertical	Fino	Código	Sistemática	Cliente (con)	Directa
Externo (público)	Caja Negra (verbatim)	Horizontal	Grueso	Diseño	Casual (ad hoc)	Servidor (para)	Indirecta
				Test...		Sistema	

Tabla 6.1. Tipos de reutilización de software

3. PAPEL DE LA MEDICIÓN EN LA REUTILIZACIÓN

Cuando la reutilización se va a llevar a cabo de forma sistemática, la primera cuestión que los gestores plantearán tendrá que ver con costes/beneficios y, a la vista de las previsiones realizadas, la política de la organización será la que decida adoptar o rechazar la reutilización. Los beneficios esperados, tanto para el proceso como para el producto, podemos resumirlos como se muestra en la tabla 6.2.

Disminución de los costes de desarrollo	Incremento de la productividad
Disminución de riesgos	Mejora de la calidad
Menor tiempo de desarrollo	Mantenimiento más fácil

Tabla 6.2. Beneficios de la reutilización

Parte de las experiencias en industrias referenciadas en [Poulin, 1996], confirman, aparentemente, esta expectación:

- La Nippon Electric Company obtuvo 6,7 veces más productividad y 2,8 veces mejor calidad con un porcentaje del 17% de reutilización.
- La GTE Corporation ahorró 14 millones de dólares durante el desarrollo de software utilizando niveles de reutilización del 14%.
- Toshiba redujo el 20-30% la tasa de defectos cuando los niveles de reutilización alcanzaron el 60%.

Si queremos sacar conclusiones válidas de todas estas experiencias deberemos tener claro:

- Qué han supuesto como reutilización.
- Cómo lo han medido.

Sólo así se podrá predecir que, siguiendo los mismos procedimientos, se obtendrán resultados similares. Esto requiere que la definición de reutilización este perfectamente establecida, además de medidas coherentes de los beneficios, que involucrarán de una u otra forma, medidas del producto y del proceso.

La medición en reutilización comporta un conjunto de problemas que son comunes a la medición de cualquier otro aspecto del software: objetivos de la medición, qué medir, cómo hacerlo, propiedades de la medición... En [Fenton et al.,1997] y [Poulin, 1996 p.10] se proponen paradigmas de la medición de software útiles en el caso de la reutilización. Este último propone como objetivos de las métricas de reutilización los siguientes:

- Que sean realistas.
- Que estimen los beneficios de reutilizar.
- Que proporcionen información a desarrolladores y gestores.
- Que proporcionen valores fáciles de entender.
- Que cumplan determinadas propiedades, como la consistencia.

Las unidades que se suelen utilizar en las diferentes métricas de la reusabilidad se expresan en Líneas De Código (LDC). Éstas presentan ciertos inconvenientes como unidades de medida, y además se refieren exclusivamente a la reutilización de componentes en la fase de implementación. Pero las LDC están más relacionadas con el esfuerzo que otras unidades diferentes, objetos, funciones... Parece razonable pensar que el esfuerzo "ahorrado" cuando se reutilizan 5 objetos de 50 KLDC cada uno es diferente al "ahorrado" cuando se reutilizan 5 objetos de 50 LDC.

Algunos de los principales modelos de métricas de reutilización y modelos económicos que podemos utilizar son:

- Modelos de nivel de reutilización, que nos sirven para estimar cuánto hemos reutilizado, diferenciando código nuevo de código reutilizado o adaptado. Las métricas sobre cuánto se ha reutilizado en una organización o aplicación intervienen en los modelos de costes-beneficios, y sirven como estimadores de los parámetros que intervienen en hipótesis sobre la mejora del proceso cuando se practica reutilización.
- Modelos de reusabilidad que describen el patrón de un componente reutilizable. Esta medida es de interés desde el punto de vista de un repositorio al que se va a incorporar el elemento reutilizable, o desde el punto de vista del desarrollo para reutilización, en el que se tratarán de mejorar y medir aquellos atributos que estén relacionados con la reusabilidad.
- Modelos de influencia de la reutilización, para estimar cuánto se ha mejorado con la reutilización. Las métricas de influencia de la reutilización son, fundamentalmente, índices de productividad, i.e, beneficios que aporta la reutilización cuando se compara con la misma acción realizada sin reutilización.
- Modelos económicos, que incluyen:

- Modelos de costes-beneficios, los cuales sirven para decidir si se debe reutilizar o no. En estos modelos deberemos disponer de estimaciones del total de costes y total de beneficios, con y sin reutilización.
- Modelos de ahorro de costes que permiten estimar cuánto dinero vamos a ahorrar si reutilizamos en vez de construir desde cero.
- Y por último modelos de recuperación de la inversión en reutilización:RIR (Return Of Investment: ROI), que permiten estimar el beneficio neto de la reutilización, después de haber consumido ciertos recursos, para justificar su uso a largo plazo.

4. MÉTRICAS DEL NIVEL DE REUTILIZACIÓN

Cuando se elaboren medidas sobre cuánto se ha reutilizado en una aplicación o en una organización, debe estar claro qué se considera como reutilización en el entorno de trabajo. En la tabla 6.3 se resumen parte de las preguntas que deben estar perfectamente contestadas para que las mediciones en una organización sean coherentes y comparables. En cualquier caso algunas se responden siguiendo las definiciones de estándares. IEEE por ejemplo sólo admite el tipo caja negra como reutilización, y en [Poulin, 1996] se encuentra una amplia discusión sobre estas cuestiones.

¿Consideramos reutilización...?	
• Mantenimiento	• Tipos parametrizados
• Sistema operativo	• Subclasificación, con polimorfismo
• Lenguajes de alto nivel	• Subclasificación, sin polimorfismo
• Herramientas	• Composición de objetos
• Usos múltiples	• Caja negra
• Bibliotecas de utilidades	• Caja blanca
	• Bibliotecas específicas de un dominio

Tabla 6.3. Cuestiones a tener en cuenta para definir reutilización

porcentaje de reutilización

Una vez que está claro que consideramos reutilización y los tipos permitidos, tenemos las bases para poder cuantificar cuánto hemos reutilizado, bien dentro de una organización o en otros niveles (equipo, aplicación). La medida que utilicemos tendrá las propiedades que caracterizan al tamaño de un elemento software. La medida estándar es el Porcentaje de Reutilización (PR) definido como:

- PR = (software reutilizado / software total) * 100.

La medida equivalente en tanto por uno es la tasa de reutilización. Esta medida es fácil de entender y calcular, pero debe estar claro que se está contando como software reutilizable y como software total. ¿Se incluye un elemento software tantas veces como se use o sólo una vez?... Se verá más adelante, en este mismo apartado, los equívocos a que puede dar lugar esta medida, dependiendo de la respuesta a estas cuestiones.

La cuantificación también puede hacerse desglosando la suma total R(S) en los diferentes niveles disjuntos que nos proporcionen los tipos permitidos. De esta forma, si llamamos R(Si) a cuanto se ha reutilizado en S de tipo i, tendremos:

- $R(S) = \sum_{i=1}^{n} R(S_i)$

Siguiendo este modelo, [Basili et al., 1996] propone una tasa para medir cuánto se ha reutilizado en el desarrollo de un sistema S, Tasa(S). Para ello tiene en cuenta:

- ✓ Las situaciones que pueden presentarse en OO con respecto a la dimensión que hemos llamado "modificación".
- ✓ La utilización de una biblioteca de elementos reutilizables, L, para desarrollar el sistema S, donde R(X) mide la cantidad reutilizada para obtener X y T(X) es el tamaño en LDC de un componente X.

Las métricas que define son las siguientes:

- R(S) = T(clase de L no modificada) + T(Clase que hereda de clase de L) + T(subclase de clase de L) + $Tasa_c$ * (T(Clase modificada))
- $Tasa_c$ = (nº de LDC de la clase que se han modificado / T(clase modificada))

En algunos casos se trabaja con valores discretos para esta tasa, asociados a una clasificación de la misma según diferentes rangos, entre otras cosas porque la estimación puntual de esta tasa puede ser más complicada que la estimación por intervalo, i.e., la tasa esta comprendida entre Po ± error [Poulin, 1996]. Hay que tener en cuenta que ancestros y subclases de C se incluyen en T(C) en determinados casos (como caja negra, herencia...), con lo cual se inflará artificialmente R(S). Esta medida se puede ajustar posteriormente en un análisis estático más detallado. La tasa de reutilización en el desarrollo de S la define como:

- Tasa (S) = R(S) / T(S)

Desde el punto de vista del cliente se pueden utilizar medidas del nivel de reutilización similares a las anteriores, y como no hay cliente sin servidor, se pueden deducir por simetría las medidas para el servidor a partir de las del cliente. En [Bieman, 1991], [Bieman et al., 1995a] y [Bieman et al., 1995b] se estudian diferentes medidas para caracterizar cada clase considerada como un cliente (servidor). Utiliza como ayudas para las métricas un grafo de llamadas y el árbol de jerarquía de herencia. Algunas de las métricas aparecen en la tabla 6.4. En [Bieman et al., 1995b] se hace una descripción más detallada de las métricas clasificadas desde la perspectiva cliente-servidor.

En [Banker et al., 1994] se implementa una colección de métricas para un repositorio utilizando como unidades de medida objetos software (módulos, macros...), sin diferenciar por tamaño. Se utilizaron, entre otras, el porcentaje de reutilización PR1 y una medida de la influencia de la reutilización (IR), que se definen como sigue:

- PR1 = (1 - (1 / IR)) * 100
- IR = (total de objetos usados / total objetos nuevos construidos)

Cliente	**Servidor**	**Sistema**

• Nº de clases reutilizadas como cajas negras • Nº de clases reutilizadas directas e indirectas • Nº de clases reutilizadas por herencia • Longitud media de los caminos	• Nº de referencias a cada clase (A_i) • Nº de subclases de A_i • Longitud media de los caminos	• % importado de la biblioteca • % de clases importadas como cajas negras • % de clases derivadas (reutilización con modificaciones) total y de la biblioteca

Tabla 6.4. Métricas cliente-servidor-sistema

En este caso contaron tanto reutilización interna como externa y además cada uso del mismo objeto. Este tipo de métricas da lugar a paradojas. Por ejemplo, en una aplicación se han desarrollado 50 objetos nuevos y se han usado 400 objetos en total. Los valores obtenidos para las métricas anteriores son: PR1 = 87,5% , IR = 8. Estas mediciones serán las mismas tanto para aplicaciones en las que 50 objetos nuevos se hubieran usado repetidamente, hasta 400 veces (en cuyo caso el porcentaje de reutilización debería ser cero), como para aplicaciones en las que se hubieran reutilizado 350 objetos exteriores a la aplicación, una vez cada uno.

5. REUSABILIDAD DE UN ELEMENTO SOFTWARE

La práctica del desarrollo con/para reutilización se plantea como un nuevo paradigma de desarrollo que permite resolver determinados problemas y aporta beneficios. El Departmento de Defensa de USA (DoD) podría ahorrar 300 millones de dólares anuales si incrementara en un 1% su nivel de reutilización [Poulin, 1996]. Pero es evidente que este tipo de desarrollo requiere elementos software que se puedan reutilizar, o mejor aún, almacenes con elementos reutilizables de calidad "garantizada" y accesibles. La base de la reutilización es el elemento o componente reutilizable, por ello parece fundamental definir que convierte a un componente en reutilizable. Esto nos permitirá saber como desarrollar componentes para los repositorios, e identificar los componentes que son potencialmente reutilizables.

Según el estándar IEEE std 1061-1992 [IEEE, 1994] la reusabilidad como factor externo de la calidad del software se define como "*el esfuerzo relativo necesario para adaptar un elemento software a otra aplicación*". Ateniéndonos a esta definición, la medición de ese esfuerzo "relativo" una vez que se ha adaptado el componente, nos podría servir para estimar la reusabilidad de éste a posteriori de su reutilización.

Entre los tipos de elementos software que forman parte de una aplicación, los específicos del dominio son los más frecuentes, lo que nos hace pensar en la necesidad de concentrar los esfuerzos de reutilización en ellos. Además, pone de manifiesto que un elemento muy reutilizable en un dominio específico, puede no serlo en otro diferente, por lo que no tiene sentido hablar de una medida "absoluta" de la reusabilidad.

Si un equipo de desarrolladores proporciona componentes reutilizables a otros equipos de la empresa, éstos serán lo suficientemente generales como para usarlos de diferentes formas y con facilidad. Si los desarrolladores para reutilización no saben cuáles son las necesidades de sus clientes potenciales, será difícil que éstos reutilicen

dichos elementos. Con esto queremos decir que la reusabilidad de un componente depende:

- De sus características internas.
- Del marco en el que se va a reutilizar.

aproximaciones a la medición de la reusabilidad

Nos podemos acercar a la medición de la reusabilidad de un elemento software desde varios puntos de vista. Uno a priori, que nos permitiría predecir la reusabilidad de un componente, y otro a posteriori, que podría basarse en la definición de la IEEE. Además, hay otros dos enfoques que podríamos llamar *cualitativo* y *empírico:*

- Desde el punto de vista cualitativo la reusabilidad se basa en el seguimiento de líneas directrices y estándares, más o menos subjetivos.
- Desde el punto de vista empírico podemos utilizar los métodos que permiten descomponer un atributo externo en atributos internos mensurables [Fenton et al., 1997], como el Factor-Criterios-Métricas (FCM), ver figura 6.1, o el Objetivo-Preguntas-Métricas (Goal-Question-Metrics: GQM). Después se necesita confirmar, mediante experimentación, que efectivamente hay relación entre las medidas de tales atributos (Xi) y una o más medidas a posteriori (Yi) del nivel de reutilización del componente software, por ejemplo, número de veces reutilizado como caja negra o esfuerzo medio para adaptarlo. Los datos observados son los que conducirán el enfoque empírico, y mediante diseño de experimentos, se podrán encontrar relaciones significativas del tipo F(X1,X2...Xn) = G(Y1,Y2...Yk), que permitirán calibrar el peso que tiene cada atributo en la reusabilidad.

Desde hace tiempo se está investigando sobre que cualidades o atributos convierten un elemento software en reutilizable, pues el núcleo del problema de la medición reside precisamente en conocer aquello que se intenta medir [Fenton et al., 1997]. Y esto debe hacerse siguiendo el método experimental. Cuando se planifican estudios para contrastar hipótesis sobre la relación que existe entre la reusabilidad de un elemento software (i.e, número de veces que se ha utilizado) y otros atributos (tamaño, complejidad, cohesión, acoplamiento...), se debe diferenciar asociación estadística de causalidad. Además, hay que tener presente que los resultados observados pueden estar contaminados de alguna manera, i.e, los desarrolladores, por desconocimiento, utilizan como principal criterio para elegir una componente para reutilizar el tamaño, pasando por alto otros atributos.

enfoque empírico

En este caso la experimentación va a ser conductora de las métricas de reusabilidad. El conjunto de métricas seleccionadas y las observaciones obtenidas pueden proporcionar información a los clientes potenciales para elegir uno u otro componente. La NATO [NATO, 1991] recomienda utilizar cuatro métricas como indicadores de la reusabilidad de un elemento software, seleccionadas después de analizar sucesivas experiencias en reutilización. Esas métricas son:

- Número de inspecciones.

- Número de reutilizaciones.
- Complejidad
- Número de informes de problemas.

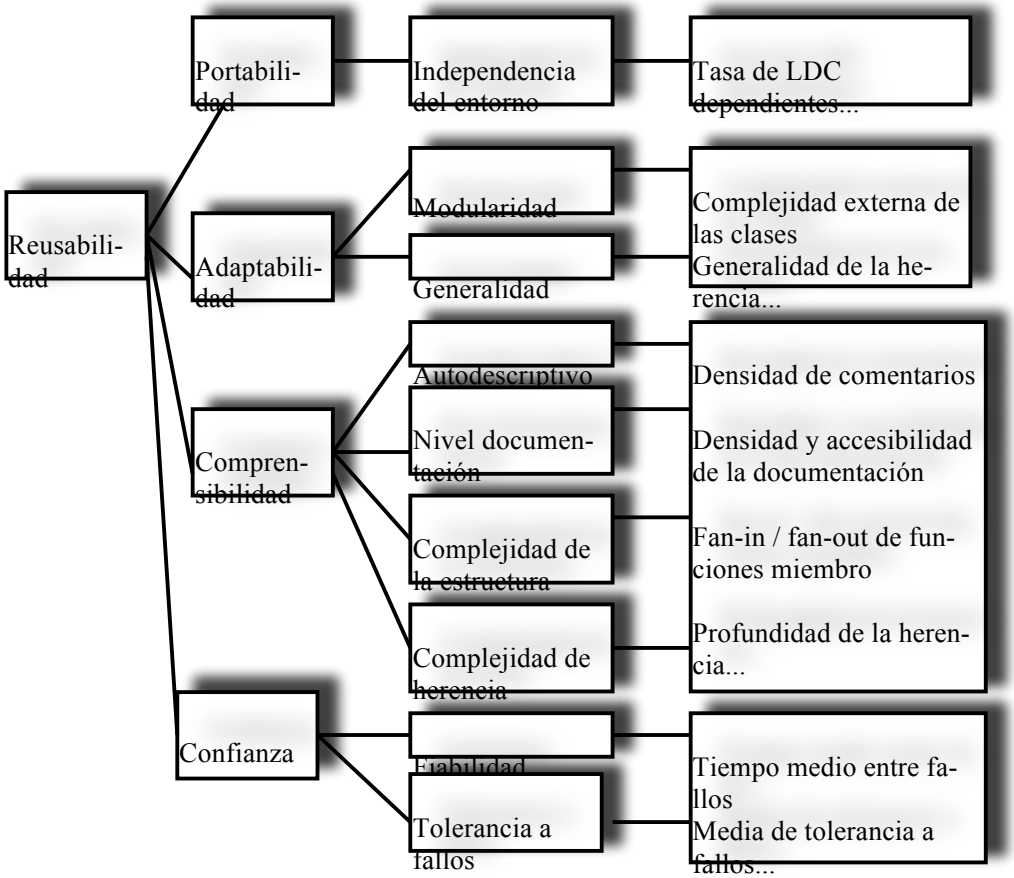

Figura 6.1. Reusabilidad según el modelo FCM, [Karlsson, 1995]

El número de inspecciones y de informes de problemas, por ejemplo, pueden ser decisivos para el cliente.

Entre los métodos empíricos se encuentra el estudio realizado en [Prieto-Díaz et al., 1987]. En dicho estudio se trataba de identificar los programas que mejor se modifican, con el fin de explotarlos en el desarrollo con reutilización de tipo caja-blanca. Se identificaron cinco atributos de programas y las medidas correspondientes, que sirvieron para evaluar la "facilidad de modificación" (siguiendo la definición de reusabilidad de IEEE). Las cinco métricas identificadas fueron:

- Tamaño del programa, medido en LDC.
- Estructura del programa medida con el acoplamiento.
- Documentación del programa.
- Lenguaje de programación.
- Experiencia en reutilización.

La experiencia, tanto en el lenguaje como en el dominio de la aplicación, condiciona los criterios que sigue el desarrollador para seleccionar los módulos a reutilizar, por lo que la métrica asociada se usó como modificador de las otras; notar que todas las métricas hacen referencia a atributos del código.

En el proyecto Reused Based on Object Oriented Techniques (REBOOT) [Karlsson, 1995] se desarrolló un modelo de reusabilidad utilizando el método FCM, junto con un modelo de calidad, que englobaba atributos no sólo de código. Los factores, criterios y métricas del modelo de reusabilidad se relacionaron como se indica en la figura 6.1. La métrica para la reusabilidad es la suma ponderada de las métricas deducidas con el método FCM, donde el peso de cada métrica debe reflejar la importancia relativa del criterio en la reusabilidad. Estos pesos se modifican en función del lugar en el que se vaya a medir la reusabilidad, dependiendo de los criterios que allí se quieran primar.

Siguiendo estos modelos, se realizó un análisis que puso de manifiesto que en la reusabilidad de los elementos software, la fiabilidad, la fácil comprensión y la facilidad para transportarlos a otro entorno tenían un peso significativo.

Otras aproximaciones a la medida de la reusabilidad en el paradigma orientado a objeto se pueden encontrar en [Price et al., 1997]. Estos autores presentan un marco para medir la reusabilidad con el fin de facilitar la reutilización de componentes en el diseño orientado a objetos. Para ello definen qué podemos reutilizar y qué no, después de clasificar las clases en generales y específicas. Además, proponen un método para construir elementos reutilizables.

enfoque cualitativo

Puesto que las aproximaciones empíricas no proporcionan una única medida de la reusabilidad, algunas organizaciones proporcionan unas guías subjetivas para identificar o desarrollar elementos software reutilizables. Es el caso de la NATO [NATO, 1991] y de REBOOT [Karlsson, 1995].

Atributo	**Descripción**
Portabilidad	El elemento software no depende de un único sistema operativo o hardware
Fiabilidad	El elemento software realiza la función de forma consistente y sin errores, superando las pruebas en diferentes entornos.
Compleción funcional	El elemento software satisface los requisitos presentes y otros que pudieran presentarse en el futuro, previsiblemente
Buen tratamiento de excepciones y errores	El elemento software documenta, trata y aísla los errores, proporcionando opciones diferentes como respuesta al error.
Ocultación de información	El elemento software es transparente al usuario, en cuanto a detalles de implementación
Fácil de comprender	El elemento software dispone de buena documentación, incluyendo comentarios en el código.

Tabla 6.5. Atributos del software reutilizable

Estas guías presentan un enfoque de ciclo de vida, abarcando desde las fases de definición y análisis de requisitos hasta la implementación, indicando el camino a seguir para diseñar o identificar software reutilizable.

En la tabla 6.5 se recogen los atributos generales de la reusabilidad de un elemento software que se mencionan en dichas guías, que no se diferencian de los principios de buena ingeniería que se deben seguir en cualquier caso.

repositorios

En el ámbito de los repositorios, la identificación de componentes reutilizables siguiendo los planes propios permitirá:

- Filtrar y cualificar los elementos que proporcionan los proveedores. La cualificación del nuevo elemento que se va a incorporar al repositorio involucra en ocasiones los dos enfoques vistos [Karlsson, 1995], [Manso et al., 1998] y las medidas de reusabilidad y calidad del elemento definirán su perfil.
- Disponer de información para decidir si el elemento software se ajusta a las necesidades del cliente.

Las diferentes aproximaciones, empíricas o cualitativas, que se pueden hacer a la reusabilidad del software tienen algunas áreas en común. La mayoría trabajan con atributos propios del código, más fáciles de medir que los de otros elementos de niveles de abstracción superior. De estos atributos, los que constituyen el común denominador de los elementos reutilizables se pueden resumir en:

- Baja complejidad.
- Buena documentación.
- Dependencias externas mínimas (bajo acoplamiento).
- Fiabilidad probada (tasa de error baja).

6. MODELOS ECONÓMICOS

Los diferentes modelos del esfuerzo, tiempo y costes-beneficios de la reutilización comprenden las perspectivas del desarrollo con y para reutilización, incluyendo en ocasiones la perspectiva del repositorio de elementos reutilizables. En [Karlsson, 1995] se detalla el conjunto de actividades que se deben incluir en la medición del esfuerzo para cada caso.

En el desarrollo para reutilización se realiza un esfuerzo extra para añadir al elemento software reutilizable:

- Reusabilidad y calidad, lo que supone actividades extra para realizar análisis del dominio, test para incrementar la fiabilidad y usos potenciales, documentación más detallada...
- Funcionalidad para generalizar y prepararlo para su distribución.

En el desarrollo con reutilización, el esfuerzo se emplea en localizar, evaluar y adaptar el componente a reutilizar.

Desde la perspectiva de un repositorio el esfuerzo es de mantenimiento: recuperación e inserción de componentes reutilizables.

costes relativos del desarrollo con y para reutilización

Los diferentes modelos económicos, que incluyen desarrollo con o para reutilización, suelen tomar como referencia los costes del desarrollo de un elemento software nuevo para un solo uso. Si este coste es C, podemos esperar que el coste de desarrollo del elemento con reutilización (CCR) cumplirá CCR ≤ C, y el del desarrollo para reutilización (CPR) cumplirá CPR ≥ C. Los costes relativos en cada caso se definen como:

- Coste Relativo de desarrollo Con Reutilización (CRCR) = CCR / C
- Coste Relativo de desarrollo Para Reutilización (CRPR) = CPR / C

El CRCR, se estima con datos históricos. Si esto se hace de forma global el error estándar puede ser considerable, para evitarlo se suelen clasificar los elementos software por complejidad, hallando estimaciones en cada estrato de la clasificación. Estudios realizados en diferentes organizaciones [Poulin, 1996 p.25], proporcionan una idea del rango del CRCR que oscila entre 0,03 y 0,40.

Obtener una estimación no sesgada y fiable del CRPR es bastante costoso, en ocasiones se proporcionan valores a esta tasa que no se basan en evidencias experimentales. Para obtener una estimación mejor se suelen clasificar los costes por tareas: costes en el análisis de dominio, requisitos adicionales, incremento de documentación, incremento de pruebas... En IBM se utilizó esta estrategia y, después de una larga experiencia en reutilización, obtuvieron una estimación del coste global adicional del 60%, así que el CRPR fue de 1,6. El rango del CRPR oscila entre 1,6 y 2,2, según los estudios anteriormente mencionados. La información que nos proporciona esta tasa puede ser más útil, como ya hemos indicado, si se clasifican los elementos reutilizables por complejidad.

coste total de un proyecto

Cuando se ha desarrollado con y/o para reutilización, el coste total del proyecto aparece desglosado en función de los tres tipos de código que se han desarrollado:

- Cantidad de código desarrollado nuevo para un solo uso (CODN).
- Cantidad de código reutilizado (CODR).
- Cantidad de código desarrollado para reutilización (CODPR).

El modelo aditivo nos proporciona la siguiente expresión del coste total, debemos tener en cuenta que para hacer una estimación del mismo necesitamos estimaciones de los parámetros que intervienen en su cálculo:

- Coste-total = (CODN + CRCR * CODR + CRPR * CODPR) * Coste-nuevo-código

modelo de Gaffney y Durek

Estos autores proponen un modelo para el indicador de la recuperación de la inversión en reutilización a largo plazo (Return Of Investment: ROI), enfocando el problema desde el punto de vista del número de reutilizaciones necesarias para amortizar la inversión. En el modelo suponen un repositorio que debe recuperar los costes invertidos en un elemento en las n primeras reutilizaciones del mismo. Además, incorporan el cambio que produce en el coste el desarrollo con reutilización, teniendo en cuenta que deben comprar un componente para reutilizar y que se va a reutilizar n veces en otros proyectos, que se ahorrarán construirlo. Consideran como referencia el desarrollo de todo el código nuevo, asignándole un valor 1; el coste C lo calculan como se indica a continuación:

- $C = (RCR + (RCPR / n)) * R + (1 - R) = R * (RCR - 1 + RCPR / n) + 1$

Donde:

RCR = coste relativo de desarrollo con reutilización
R = proporción de código reutilizado
RCPR = coste relativo de desarrollo para reutilización
n = número de reutilizaciones previstas
C = coste de desarrollo

El umbral de amortización es el número de veces que la organización debe reutilizar un componente para amortizarlo. Si en la ecuación del coste suponemos RCR y RCPR estimados, podemos hallar n para que el desarrollo con reutilización compense, o lo que es igual:

- $(RCR - 1 + RCPR / n) \leq 0 \Rightarrow n \geq RCPR / (1 - RCR)$

Si se disponen de las estimaciones siguientes: RCR = 0,3 y RCPR = 1,5, el coste será $C = (0,3 + (1,5 / n) - 1) * R + 1$, el número de reutilizaciones para amortizar el componente será mayor igual que 1,5 / (1 – 0,3). Es decir, el elemento software deberá reutilizarse al menos 3 veces.

modelo de Barnes y Bollinger

El modelo que proponen se basa en la relación que existe entre beneficios (B) e inversión en reutilización (R):

- $Q = B / R$, si $Q < 1$ la reutilización produce pérdidas
- $B = \sum_{i=1}^{n} (D_i - A_i) - I$

Donde:

D_i = coste del proyecto sin reutilización
A_i = coste neto de reutilizar, incluye el coste de encontrar, modificar e integrar el componente reutilizable

I = inversión total en reutilización, coste de producir componentes reutilizables o costes de generalización
n = número de proyectos que reutilizan el software

Pero en este caso no proponen una guía que permita clasificar las entradas en beneficios o en inversiones. En [Poulin, 1996 pp.15-16] se proponen guías que podían servir como conductoras en el cálculo de R y B. Como en el modelo anterior, podemos encontrar n para que se cumpla $B \geq 0$. Henderson-Sellers [Henderson-Sellers, 1993] plantea un modelo similar para una biblioteca de clases.

modelo de Poulin y Caruso

Poulin y Caruso [Poulin, 1996] proponen dos modelos de recuperación de la inversión en reutilización (ROI) para IBM en 1992, así como dos métricas de costes, pudíendose obtener una nueva versión de los modelos de costes–beneficios en [Poulin, 1997].

Una de las métricas propuestas se basa en el cálculo del coste que se evita con la reutilización (CER) como suma del ahorro en el desarrollo (CAD) más el ahorro por servicios evitados en la detección y eliminación de errores (CAS), en los que interviene el nivel de reutilización entre otros parámetros.

- CER = CAD + CAS = LDCR * (1 - CRCR) * CNC + LDCR * TE * CE

Donde :

LDCR = Líneas de código reutilizadas
CRCR = Coste relativo de desarrollo con reutilización
CNC = Coste de desarrollo de nuevo código
TE = Tasa de error
CE = Coste por error

La otra métrica corresponde al enfoque del servidor o proveedor, y cuantifica el valor añadido al reutilizar (VAR). Tiene un significado similar a un índice de productividad, con valor 1 para indicar una productividad normal. En ella interviene lo que se llaman instrucciones reutilizadas por otros (IRPO), parámetro que se incorpora a la producción total para el cálculo de VAR:

- $IRPO = \sum_{i} (LDC_i * NUMO_i)$

Donde LDC_i es el tamaño del componente iésimo que es utilizado por $NUMO_i$ partes. Por ejemplo, un equipo escribe un módulo de 5 KLDC que es reutilizado en la actualidad por 7 departamentos y uno de 15 KLDC reutilizado por 6 departamentos, el IRPO se calcularía: IRPO-actual = 5 KLDC * 7 + 15 KLDC * 6 = 125 KLDC.

El valor añadido al reutilizar se define como:

- VAR = (LDCT + IRPO) / (LDCT - LDCR)

Donde LDCT es el número de líneas de código desarrolladas en total. Siguiendo con el ejemplo anterior, si en el grupo se han desarrollado 200 KLDC en total y se han reutilizado 50 KLDC, se tendrá como VAR:

VAR = (200 KLDC + 125 KLDC) / (200 KLDC – 50 KLDC) = 2,17.

Este valor indicaría un incremento del 117% en la efectividad del grupo debido a la reutilización. Nótese que VAR será 1 cuando se reutilice tanto como se proporcione a otros para reutilizar.

modelo COCOMO

El modelo COCOMO básico presupone una estimación inicial del esfuerzo en KLDC, a partir de la cual proporciona predicciones del esfuerzo en tiempo de trabajo y costes. Este modelo se desarrolla con tres expresiones que corresponden a tres modalidades, básica, intermedia o detallada e identifica tres niveles de complejidad de software que influyen en la estimación que el modelo proporciona. En el COCOMO II se incorpora el efecto de la reutilización [Boehm, 1997].

Balda y Gustafson proponen una modificación al modelo del tiempo, incorporando el esfuerzo adicional cuando se desarrolla para reutilización y el ahorro al desarrollar con reutilización, con o sin modificaciones. El modelo básico se transforma con las modificaciones en la expresión siguiente:

- $T = \sum_{i=1}^{4} \left(a_i * N_i^b \right)$

N_1 = Líneas de código, en miles, desarrolladas para un solo uso (tipo 1)
N_2 = Líneas de código, en miles, desarrolladas para reutilizar (tipo 2)
N_3 = Líneas de código, en miles, reutilizadas sin modificación (tipo 3)
N_4 = Líneas de código, en miles, reutilizadas modificándolas (tipo 4)
a_i = Coste de desarrollar o adaptar las líneas del tipo i

La expresión anterior se puede simplificar cambiando los coeficientes a_i por combinaciones de los costes relativos CRCR y CRPR. Las relaciones entre los coeficientes son $a_2 = (CRPR / CRCR) * a_3$. Es decir, si CRCR = 0,3 y CRPR = 2,1 entonces desarrollar para reutilización cuesta 7 veces más que desarrollar con reutilización. Si cada a_i se pone en función de a_1, la expresión queda como sigue:

- $T = \alpha * N_1^{\beta} + (CRPR / CRCR) * \gamma * \alpha * N_2^{\beta} + \gamma * \alpha * N_3^{\beta}$

Karlsson modifica el modelo COCOMO para los dos tipos de desarrollo, con y para reutilización. Las modificaciones que plantea en el desarrollo para reutilización se basan en añadir factores de coste al modelo COCOMO básico:

- $Coste = a * KLDC^b * \prod f_i$

Donde a y b dependen del modo de desarrollo y están calibrados. Para ello tiene en cuenta el modelo de reusabilidad que ha planteado (figura 6.1) y el modelo de calidad siguiendo el método FCM. Las modificaciones se hacen teniendo en cuenta los criterios como sigue:

- Se añaden factores de costes por la independencia del entorno, fiabilidad, modularidad y simplicidad, utilizando las tablas del COCOMO.
- Se añade código extra por los criterios de documentación extra, autodescriptividad y tolerancia a fallos. La cantidad de código que se añade se hace con supuestos subjetivos en los que intervienen estimaciones medias.
- No se tienen en cuenta el resto de criterios, bien por que el coste que puede suponer frente al coste total sea despreciable o bien porque ya se han tenido en cuenta en las líneas de código extra estimadas, como es el caso de la generalidad.

Karlsson propone además una modificación del modelo COCOMO de costes para el desarrollo con reutilización, para lo cual transforma la estimación de las líneas de código que se van a reutilizar en el equivalente de líneas de código a desarrollar, y sobre ellas aplica el modelo de costes de COCOMO.

7. CONCLUSIONES

Si algo no se puede medir, entonces no se puede controlar ni tener la certidumbre de que se está caminando hacia los objetivos marcados. Si el desarrollo del software quiere alejarse del mundo de la artesanía para entrar en una era industrial, debe hacer acopio de esta máxima, y los procesos de la Ingeniería del Software deben incluir las actividades de medida necesarias para caminar hacia ese software de calidad que tanto se menciona en la bibliografía.

La reutilización no puede ser ajena a todo ello, principalmente porque nos promete beneficios a cambio de una inversión económica, y debemos disponer de instrumentos que nos permitan comprobar si dicha inversión es rentable.

Con este capítulo se ha pretendido hacer un breve repaso por los modelos de métricas más difundidos en el campo de la reutilización del software, sin llegar a ser exhaustivos. Se ha incidido especialmente en algunas de las aproximaciones a los métodos de medida de la reutilización en orientación al objeto, tratando de mostrar las diferentes facetas que envuelven al desarrollo con y para reutilización.

Todos los modelos y métricas planteados tienen básicamente el mismo objetivo: conocer qué es la reusabilidad y su influencia en el proceso de desarrollo. Las métricas deben definirse en función de los objetivos y bajo las condiciones que la teoría de la medición establece. El conocimiento se establece mediante un proceso de retroalimentación entre los modelos teóricos y las observaciones empíricas; y el diseño de experimentos debe conducir esta retroalimentación.

En este repaso también han quedado pendientes aspectos muy relacionados con la reutilización, como puede ser la reutilización en el marco del CMM (Capability Matu-

re Model) [Griss, 1998], o las herramientas que apoyan y automatizan las mediciones en el proceso de reutilización, o los aspectos organizativos o de gestión tan vitales para el establecimiento de un modelo de reutilización sistemática en una empresa.

8. BIBLIOGRAFÍA

1. [Banker et al., 1994] Banker, R.D., Kauffman, R.J., Wright, C. & Zweig, D. "Automating Output Size and Reuse Metrics in a Repository-Based Computer-Aided Software Engineering (CASE) Environment". IEEE Transactions on Software Engineering, Vol. 20, Nº 3, March 1994.
2. [Basili et al., 1996] Basili, R.V., Briand, L.C. & Melo, W.L. "Assessing the Impact of Reuse on Quality and Productivity in Object-Oriented Systems". Communications of the ACM, October 1996.
3. [Bieman, 1991] Bieman, J.M. "Deriving Measures of Software Reuse in Object Oriented Systems". Technical Report CS-91-112, July 1991.
4. [Bieman et al., 1995a] Bieman, J.M. & Zhao, J.X. "Reuse through Inheritance: A Quantitative Study of C++ Software". Proceedings of the ACM Symposium on Software Reusability, April 1995.
5. [Bieman et al., 1995b] Bieman, J.M. & Khang, B. "Cohesion and Reuse through in an Object-Oriented System". Proceedings of the ACM Symposium on Software Reusability, April 1995.
6. [Boehm, 1997] Boehm, B. "COCOMO II Model Definition Manual". Version 1.4 University of Southern California, 1997.
7. [Fenton et al., 1997] Fenton E. & Pfleeger, S.L. "Software Metrics. A Rigurous and Practical Approach". Second edition. International Thompson Computer Press, 1997.
8. [Frakes et al., 1996] Frakes, W. & Terry, C. "Software Reuse: Metrics and Models". ACM Computing Surveys. Vol. 28, Nº 2. June 1996.
9. [Freeman, 1983] Freeman, P. "Reusable Software Engineering: Concepts and Research Directions". Proceedings of the workshop on reusability in programming, 1983.
10. [García et al., 1999] García Peñalvo, F.J., Romay Rodríguez, Mª P., Marqués Corral, J.M. y Crespo González-Carvajal, Y. "*Mecanos: Exposición de Resultados y Líneas de Trabajo Abiertas en la Reutilización Sistemática del Software*". En las actas de las 2as Jornadas Iberoamericanas de la Ingeniería de Requisitos y Ambientes de Software IDEAS'99 (Alajuela-Costa Rica, 24-26 de Marzo 1999): 193-204. 1999.
11. [Griss, 1998] Griss M.L. "Reuse Strategies. CMM as a Framework for Adopting Systematic Reuse". Object magazine, March 1998.
12. [Henderson-Sellers, 1993] Henderson-Sellers, B. "The Economics of Reusing Library Classes". Journal of Object-Oriented Programming, Vol. 6, Nº 4. 1993.
13. [IEEE, 1994] IEEE. "Standards Collection Software Engineering". IEEE, 1994.
14. [Jacobson et al., 1997] Jacobson I., Griss, M.L. & Jonsson, P. "Software Reuse. Architecture, Process and Organization for Business Success". ACM Press. Addison Wesley Longman, 1997.
15. [Karlsson, 1995] Karlsson E. (editor). "Software Reuse. A Holistic Approach", John Wiley & Son Ltd. 1995.

16. [Manso et al., 1998] Manso, Mª E., Romay, Mª P. & García, F.J. "Modelo de Cualificación de Assets del Repositorio GIRO". III Jornadas de Trabajo de MENHIR. Moros, B. & Sáez, J. (editores). Murcia, Noviembre 1998.
17. [McClure, 1997] McClure, C. "Software Reuse Techniques. Adding Reuse to the Systems Development Process". Prentice Hall, 1997.
18. [NATO, 1991] NATO. "Standard for the Development of Reusable Software Components". NATO Communications and information systems agency, 1991.
19. [Price et al., 1997] Price, M.W. & Demurgian, S.A. "Analyzing and Measuring Reusability in Object-Oriented Designs". ACM 0-89791-908-4/97 0010, 1997.
20. [Prieto-Díaz, 1993] Prieto-Díaz, R. "Status report: Software reusability". IEEE Software. May 1993.
21. [Prieto-Díaz et al., 1987] Prieto-Díaz, R. & Freeman, P. "Classifying Software for Reusability". IEEE Software, Vol 4, Nº 1:6-16, January 1987.
22. [Poulin, 1996] Poulin, S.J. "Measuring Software Reuse". England. Addison-Wesley, 1996.
23. [Poulin, 1997] Poulin, S.J. "Fueling Software Reuse". Object magazine, September, 1997.