V Jornadas de trabajo de MENHIR

Granada, 30-31 Marzo del 2000

ACTAS

Organizado por:

GEDES
Grupo de Especificación Diseño y Evolución del Software

Colaboradores:

Universidad de Granada

E.T.S.I. Informática

Dpto. Lenguajes y Sistemas Informáticos

Comité de organización:

Ana Anaya Morito
Mª Visitación Hurtado Torres
Fernando Molina Ortiz
Patricia Paderewski Rodríguez
José Parets Llorca
Mª José Rodríguez Fórtiz

Subvencionado por:

Proyecto MENHIR (CICYT TIC 97-0593-C05-04)

Editado por:

Mª José Rodríguez Fórtiz
Patricia Paderewski Rodríguez

INDICE

Una propuesta para medir la calidad de los diagramas de clases en UML

Marcela Genero [•], M[a] Esperanza Manso[✦], Mario Piattini [•]

[•] Departamento de Informática - Universidad de Castilla-La Mancha
Tel. (+34) 926 29 53 00 Ext. 3715 – Fax (+34) 926 29 53 54
Ciudad Real – Spain
{mgenero, mpiattin}@inf-cr.uclm.es

[✦]Departamento de Informática - Universidad de Valladolid
Tel. (+34) 983 42 30 00 – Fax 983 42 36 71
Valladolid – Spain
manso@infor.uva.es

Resumen

La calidad del desarrollo de software orientado a objetos (OO), depende en gran medida de los productos obtenidos en etapas tempranas del ciclo de vida. Por ello es necesario evaluar la calidad de los diferentes productos obtenidos en la etapas de especificación de requisitos y análisis. La "complejidad" tiene una gran influencia en los distintos factores que influyen el la calidad de cualquier producto de software, como por ejemplo en los diagramas de clase. Por ello, en este trabajo proponemos y validamos empíricamente un conjunto de métricas para medir la complejidad de los diagramas de clases especificado utilizando el Lenguaje Unificado de Modelado (UML). La principal ventaja de las métricas propuestas, con respecto a las ya existentes, es que pueden aplicarse en las etapas tempranas del ciclo de vida del sistema que se está desarrollando, pudiendo de esta forma analizar el mismo a medida que se va desarrollando, permitiendo así realizar mejoras antes de llegar a la implementación.

1. Introducción

El paradigma OO ha surgido como una solución para el diseño y desarrollo de software, y su adopción ha ido imponiendo en la industria. En los últimos años ha habido una explosión de propuestas de metodologías de análisis y diseño, lenguajes, sistemas de gestión de bases de datos basados en el paradigma de la orientación a objetos (OO). El Lenguaje de Modelado Unificado(UML) propuesto por Booch et al. (1998) ha surgido como un lenguaje estándar para expresar los distintos modelos necesarios para diseñar software OO. El hecho de que exista un nuevo estándar de modelado, no asegura la calidad de los modelos producidos a largo del proceso de desarrollo de software OO, por lo tanto es necesario evaluar tal calidad.

La calidad del software es un término vago y polifacético, ya que puede tener distinto significado para diferentes personas. Para paliar este problema ha surgido un estándar para calidad de producto de software, ISO 9126 (1999), que propone un modelo de calidad. Este modelo define una jerarquía de factores que influyen en la calidad, como por ejemplo funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y trasportabilidad. A su vez cada uno de estos factores se subdividen en otros.

Sin embargo contar con una lista de criterios de calidad, no es suficiente para asegurar la calidad en la práctica, porque diferentes personas generalmente tienen diferentes interpretaciones sobre un mismo concepto. Es necesario establecer métricas para valorar los diferentes factores que influyen en la calidad, es decir es necesario contar con "criterios medibles" para evitar las tendencias subjetivas (Zultner, 1992). De hecho, actualmente se reconoce que la medición del software es el medio más efectivo para comprender, monitorizar, controlar, predecir y mejorar el desarrollo del software (Briand et al., 1996a). La medición no se utiliza sólo para comprender y mejorar el desarrollo, sino también para determinar la mejor manera de ayudar a los profesionales y a los investigadores

(Pflegger, 1997).

El surgimiento del desarrollo OO ha tenido un inconveniente adicional para medir la calidad del software, las métricas tradicionales de productos de software no necesariamente se aplican tan bien al desarrollo OO, debido a algunas nociones fundamentales (encapsulación, herencia, polimorfismo) inherentes al desarrollo de software OO. Como mínimo, a estas métricas tradicionales se deben ser añadir métricas que sean específicas y apropiadas para el marco del desarrollo de software OO.

Una complicación adicional es que la mayoría de las métricas disponibles actualmente están enfocadas en medir aspectos referidos a la etapa de diseño eimplementación (Brito e Abreu y Carapuça, 1994; Chidamber y Kemerer, 1994; Fenton y Pfleeger, 1997, Henderson-Sellers, 1996), dejando un poco de lado aspectos referidos a la etapa de análisis. Estas métricas proveen información demasiado tarde para ayudar a mejorar las características internas del producto antes de su finalización. Por eso hay una gran necesidad de contar con métricas que puedan aplicarse en las etapas tempranas del desarrollo, particularmente durante la especificación de requerimientos y el análisis, para asegurar que los productos obtenidos en la etapa de análisis tienen propiedades internas favorables que pueden conducir a desarrollar productos de software de calidad.

Este enfoque de la medición puede brindarles a los desarrolladores una oportunidad para solucionar problemas, eliminar atributos de diseño no correctos, y eliminar la complejidad no deseada muy temprano en el ciclo de desarrollo. Esto reduciría considerablemente el trabajo extra necesario durante las etapas de implementación y posterior mantenimiento.

La mantenibilidad es uno de los factores que influyen en la calidad de un producto software (ISO 9126, 1999) y se encuentra altamente influenciada por la complejidad de su diseño (Li and Cheng, 1987). Al mencionar término complejidad, nos referimos a la "complejidad del producto", también llamada complejidad estructural (Henderson- Sellers, 1996).

El objetivo de este trabajo es medir la complejidad de los diagramas de clases construidos utilizando el estándar UML (Booch et al., 1998). Es importante tener en cuenta que es imposible obtener una única medida global de la complejidad (Fenton, 1994). Por ello, en este trabajo presentaremos un conjunto de métricas (sección 2) que permitan evaluar la complejidad de los diagramas de clases, midiendo sus diferentes componentes (atributos, relaciones, clases, etc.). En la sección 3 realizaremos la validación empírica de las métricas, aplicándolas a diagramas de clases del repositorio GIRO (Grupo de Investigación en Reutilización y Orientación al Objeto). Y finalmente en la sección 4 presentamos nuestras conclusiones como así también las líneas de trabajo futuro.

2. Métricas para diagramas de clases en UML

En esta sección presentamos un conjunto de métricas "acotadas" (Lethbridge, 1998) para medir la complejidad de los diagramas de clases. Una métrica acotada es aquella en la cual las mediciones siempre caen dentro de un rango (en nuestro caso [0,1]), y es imposible que sobrepasen dicho rango. Por ejemplo, la mayoría de los valores de las métricas que propondremos caen en el rango [0,1].

Las métricas que definiremos a continuación surgieron como adaptación a UML de otras propuestas en (Genero et al., 1999) referidas a diagramas de clases en OMT.

Estas métricas están basadas en la teoría de la complejidad, que define a la complejidad de un sistema como el número de componentes del sistema y el número de relaciones entre los componentes. Como la meta es simplificar el diagrama de clases tanto como sea posible, el objetivo será minimizar el valor

de estas métricas. Consideramos que si el valor de una métrica tiende a cero es el mejor caso, y si tiende a uno el peor caso.

En primer lugar consideraremos los siguientes tipos de relaciones que se pueden especificar en UML: asociaciones, agregaciones, generalizaciones y dependencias.

2.1 Métrica ASvsC

La métrica Asociaciones vs. Clases mide la relación que existe entre el número de asociaciones y el número de clases en un diagrama de clases en UML. Esta métrica esta basada en la métrica M_{RPROP} propuesta por Lethbridge, T. (1998).

Definimos esta métrica de la siguiente manera:

$ASvsC = \left(\frac{N^{AS}}{N^{AS} + N^{C}} \right)^2$	N^{AS} es el número de asociaciones en un diagrama de clases. N^{C} es el número de clases en un diagrama de clases. Siendo $N^{AS} + N^{C} > 0$.

2.2 Métrica N-aryAss

La métrica Asociaciones N-arias mide el número de relaciones N-arias (no binarias) comparado con el número total de asociaciones en un diagrama de clases en UML.

Definimos esta métrica de la siguiente manera:

$N-aryAss = \frac{N^{N-aryAss}}{N^{Ass}}$	$N^{N\text{-}aryAss}$ es el número de asociaciones N-arias en un diagrama de clases. N^{Ass} es el número de asociaciones en un diagrama de clases. Siendo $N^{Ass} > 0$.

2.3 Métrica AGvsC

La métrica Agregaciones vs. Clases mide la relación que existe entre el número de agregaciones y el número de clases en un diagrama de clases en UML.

Definimos esta métrica de la siguiente manera:

$AGvsC = \left(\frac{N^{AG}}{N^{AG} + N^{C}} \right)^2$	N^{AG} es el número de agregaciones en un diagrama de clases en UML N^{C} es el número de clases en un diagrama de clases. Siendo $N^{AG} + N^{C} > 0$

Consideramos como el número de agregaciones cada nivel de la jerarquía de agregación, es decir, cada símbolo $\Diamond$ en el diagrama de clases.

2.4 Métrica GEvsC

La métrica Generalizaciones vs. Clases mide la relación que existe entre el número de jerarquías de generalización y el número de clases que forman parte de tales jerarquías, en un diagrama de clases UML.

Definimos esta métrica de la siguiente manera:

$GEvsC = \left(\frac{N^{GE}}{N^{GE} + N^{C}} \right)^2$	N^{GE} es el número de jerarquías de generalización en un diagrama de clases N^{C} es el número de clases en el diagrama de clases que forman parte de las jerarquías de generalización Siendo $N^{GE} + N^{C} > 0$.

Consideramos como el número de generalizaciones, cada nivel de la jerarquía de generalización, es decir cada símbolo △en el diagrama de clases.

2.5 Métrica M_{JG}

El objetivo de la métrica Jerarquía de Generalización (M_{JG}) es evaluar la complejidad introducida por las jerarquías de generalización en un diagrama de clases, teniendo en cuenta algunos factores que influyen en el "tamaño de una jerarquía" (cantidad de clases, cantidad niveles en la jerarquía y sobre todo la herencia múltiple).

M_{JG} se define de la siguiente manera:

1. Si el diagrama de clases *no tiene* jerarquías de generalización: $M_{JG} = 0$
2. Si el diagrama de clases *tiene* jerarquías de generalización M_{JG} se define de la siguiente manera:

   $M_{JG} = \sum_{i=1}^{n} CJ_i$, siendo CJ_i la complejidad de la i-ésima jerarquía de generalización y n el número total de jerarquías de generalización en un diagrama de clases.

Para calcular CJ_i se combinan dos factores. El primer factor es la fracción de entidades en la jerarquía de generalización, que son hojas. Esta medida, llamada $FHOJA_i$ se calcula así: $FHOJA_i = \frac{N_i^{HOJA}}{N_i^{C}}$, siendo N_i^{HOJA} el número de clases que son hojas en la i-ésima jerarquía de generalización, y N_i^{C} el número total de clases en la i-ésima jerarquía de generalización..

La figura 1 muestra diferentes jerarquías de generalización con sus respectivos valores de $FHOJA_i$. $FHOJA_i$ tiende al valor 0,5 cuando el número de hojas es la mitad del número de entidades (figura 1, ej. c y d). Tiende a cero en el caso de un árbol unario (figura 1, ej. b). Y tiende a 1 cuando cada clase es una subclase de la superclase de la jerarquía (figura 1, ej. a). El factor $FHOJA_i$ tiene una propiedad poco deseable, para jerarquías poco profundas (dos o tres niveles) con una gran cantidad de hojas nos da un valor demasiado alto (figura 1, ej. a). Para corregir este problema, utilizamos un factor adicional para calcular el valor de la métrica CJ_i, el promedio de las subclases directas o indirectas para cada subclase que no sea la superclase de la jerarquía. A este factor lo denominamos $ALLSUP_i$ (la superclase de la jerarquía no la tenemos en cuenta ya que no tiene padres).

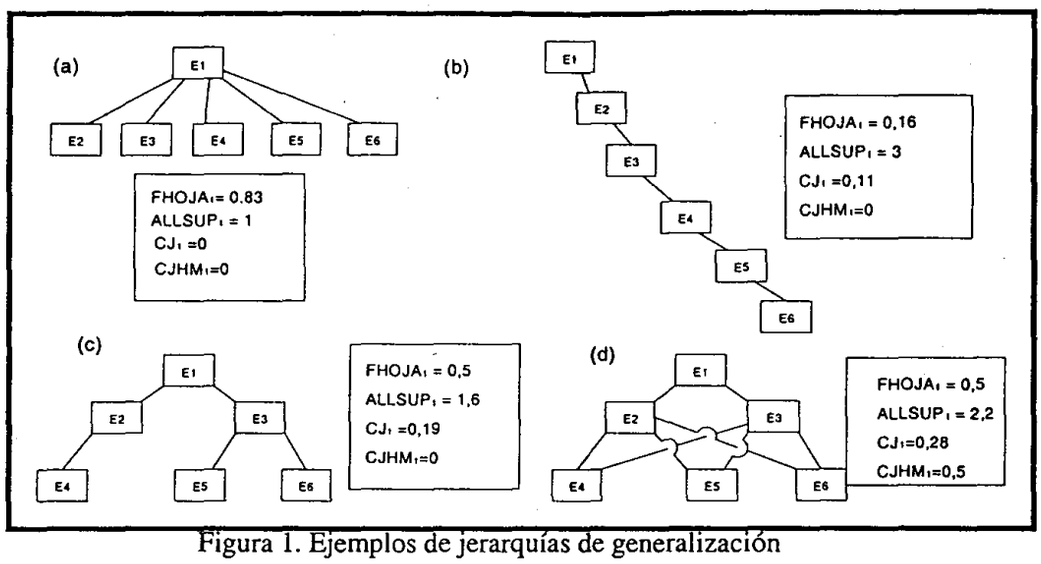

Figura 1. Ejemplos de jerarquías de generalización

Por lo tanto podemos definir CJ_i de la siguiente manera: $CJ_i = FHOJA_i - \frac{FHOJA_i}{ALLSUP_i}$

2.6 Métrica M_{HMJG}

La métrica Herencia Múltiple para la Jerarquías de Generalización (M_{HMJG}) evalúa la complejidad introducida por la herencia múltiple en las jerarquías de generalización en un diagrama de clases.

La experiencia ha demostrado que el uso de herencia múltiple puede llevar a complicaciones imprevistas, y por eso es conveniente evitarla siempre que sea posible.

M_{HMJG} se define de la siguiente manera:
1. Si el diagrama de clases *no tiene* herencia múltiple $M_{HMJG}= 0$

Si el diagrama de clases *tiene* jerarquías de generalización MHMJG se define de la siguiente manera:

$M_{HMJG} = \sum_{i=1}^{n} CJHM_i$, siendo $CJHM_i$ la complejidad de herencia múltiple de la i-ésima jerarquía de generalización, y n es el número total de jerarquías de generalización en un diagrama de clases.

$CJHM_i$ mide el ratio de padres extras (más de una superclase) que tiene cada clase y se calcula de la siguiente manera: $CJHM_i = \frac{N_i^{EX}}{N_i^C}$, siendo N_i^{EX} el número de superclases (padres) extras que tienen las clases en la i-ésima jerarquía , y N_i^C es el número total de clases en dicha jerarquía de generalización.

2.7 Métrica DEPvsC

La métrica Dependencias vs. Clases mide la relación que existe entre el número de relaciones de dependencia y el número de clases en un diagrama de clases en UML.

Definimos esta métrica de la siguiente manera:

$DEPvsC = \left(\frac{N^{DEP}}{N^{DEP} + N^{C}} \right)^2$	N^{DEP} es el número de relaciones de dependencia en un diagrama de clases N^{C} es el número de clases en un diagrama de clases Siendo $N^{DEP} + N^{C} > 0$.

2.8 Métrica AvsC

La métrica Atributos vs. Clases mide la relación que existe entre el número de atributos y el número de clases en un diagrama de clases en UML.

Definimos esta métrica de la siguiente manera:

$AvsC = \left(\frac{N^{A}}{N^{A} + N^{C}} \right)^2$	N^{A} es el número de atributos en un diagrama de clases N^{C} es el número de clases en un diagrama de clases Siendo $N^{A} + N^{C} > 0$

2.9 Métrica MEvsC

La métrica Métodos vs. Clases mide la relación que existe entre el número de métodos y el número de clases en UN diagrama de clases en UML.

Definimos esta métrica de la siguiente manera:

$MEvsC = \left(\frac{N^{ME}}{N^{ME} + N^{C}} \right)^2$	N^{ME} es el número de métodos en un diagrama de clases N^{C} es el número de clases en un diagrama de clases Siendo $N^{ME} + N^{C} > 0$

3. Validación empírica de las métricas

El repositorio GIRO (Grupo de Investigación en Reutilización y Orientación al Objeto) incluye elementos reutilizables o assets procedentes de cualquier fase del proceso de desarrollo y de cualquier paradigma, en particular OO o estructurado (García F.J., Marqués, J.M. y Maudes, J.M. ,1997; García F.J., Marqués, J.M., Laguna M.A. y Maudes, J.M., 1998; Manso E, García F.J., Rodríguez J.J, Laguna M.A., 1999).

Los elementos del repositorio que serán objeto de estudio son diagramas de clases que pertenecen a tres líneas de producto: Imagen (I), Discapacitados (D) y Ópticas (O). Todos ellos se desarrollaron siguiendo el paradigma OO y UML como lenguaje de modelado, utilizando la herramienta RATIONAL ROSE para el análisis-diseño.

La estrategia que hemos seguido para aplicar las métricas propuestas con anterioridad a los assets seleccionados consiste en estudiar

1) Estimaciones que resumen las mediciones observadas
2) Conjunto de métricas que pueden diferenciar las líneas de producto
3) Posibles relaciones entre las diferentes métricas

Las técnicas de análisis estadístico utilizadas sirven para estudiar variables aleatorias ordinales, pues hemos supuesto que la escala de las métricas es al menos ordinal.

3.1 Estimaciones de las mediciones observadas

La tabla 1 resume las mediciones observadas en cada línea de producto y totales, para cada uno de ellos la primera fila son totales y la segunda los promedios. Y los gráficos de Kiviat de las figuras 2-5 presentan los resultados totales y por línea de producto. Se puede observar que la mayor "complejidad", (métricas próximas a 1) la aportan el nº de atributos y métodos, si comparamos el valor de los percentiles frente a los de las otras métricas.

	nºde assets	ASvsC	N-ary	AGvsC	GEvsC	DEPvsC	AvsC	MEvsC	Mjg
D	7	0,1180				0,0710	0,3690	0,6521	
I	7	0,0170		0,0255	0,0481		0,5390	0,6260	0,1100
O	13	0,0641		0,0085	0,1059		0,6615	0,6450	0,0556
T (totales)	27	0,1061		0,0183	0,0773	0,0394	0,8220	1,0204	0,0755
Percentil 25		0,0123					**0,4970**	**0,6463**	
Percentil 50		0,0512					**0,5710**	**0,6694**	0,055
Percentil 75		0,1182		0,0052		0,0400	**0,6690**	**0,7137**	0,385

Tabla 1. Resultados de los valores obtenidos

3.2 Utilización de las métricas para analizar el conjunto de métricas que pueden diferenciar las líneas de producto

Puesto que se ha trabajado con líneas de producto diferentes, es razonable estudiar si las métricas de complejidad en estudio se comportan de forma distinta en los 3 grupos. Evidentemente la muestra no es muy grande, pero aún así, nos pueden dar una idea de, hasta que punto, las líneas de producto son diferentes. Este estudio lo hemos realizado utilizando el test de Mann-Whitney para comparación de medianas , puesto que las variables en estudio se suponen de escala ordinal al menos (tabla 2).

	AsvsC	AgvsC	GEvsC	DEPvsC	AvsC	MEvsC	Mjg
I/D	0.0087		0.9350		0.2013	0.7983	
I/O	0.0383	0.0066			0.0383	0.4272	
D/O	0.1303				0.0015	0.5253	

Tabla 2. p-valor para el Test de significación de Mann-Whitney

A la vista de estos resultados parece que:

- La línea de producto D se diferencia de la de O por la métrica AvsC, y de la de I por la métrica AsvsC.
- La línea de producto I se diferencia de la de O por las métricas ASvsC, AGvsC y AvsC.
- La métrica MEvsC no diferencia las líneas de producto (esta métrica es similar en las tres).

3.3 Análisis sobre la independencia de la métricas

El test de correlación de Spearman nos permitirá estudiar el grado de independencia de las métricas, o hasta que punto la información que nos proporcionan puede ser redundante. En las tablas 3,4 y 5 se presentan 3 valores para cada métrica y línea, que corresponden al p-valor, nº de pares y estimación del coeficiente de correlación (Rs) de Spearman. La Hipótesis del contraste es que no hay dependencia (Ho: Rs = 0)

	ASvsC	AGvsC	GEvsC	DEPvsC	AvsC	MEvsC
ASvsC		0.7874 7 -0.1101	0.9580 5 -0.0263		0.7574 7 -01261	0.4017 7 -0.3424
AGvsC			0.1944 5 -0.6489		0.1088 7 0.6547	0.0325 7 0.8729
GEvsC					0.4726 5 0.3591	0.8374 5 -0.1026
DEPvsC						
AvsC						0.0662 7 0.75

Tabla 3. Correlación en la línea de Imagen

	ASvsC	AgvsC	GEvsC	DEPvsC	AvsC	MEvsC
ASvsC				0.6273 7 0.1982	0.1153 7 0.6429	0.7930 7 0.1071
AGvsC						
GEvsC						
DEPvsC					0.3315 7 0.3964	0.7574 7 0.1261
AvsC						0.7930 7 -0.1071

Tabla 4. Correlación de las métricas en la línea de Discapacitados

	AsvsC	AGvsC	GEvsC	DEPvsC	AvsC	MEvsC
ASvsC		0.2452 13 0.3354	0.8493 10 -0.0633		0.2156 13 0.3574	0.1507 13 0.4148
AGvsC			0.8067 10 -0.0815		0.1619 13 -0.4038	0.9068 13 -0.0338
GEvsC					0.5828 10 0.1831	0.0154 10 -0.8074
DEPvsC						
AvsC						0.7366 13 -0.0971

Tabla 5. Correlación de las métricas en la línea de Ópticas

A partir de los resultados de las tablas 3, 4 y 5, parece que:

- En la línea de producto I, pueden estar relacionadas las métricas AGvsC y MEvsC (ver tabla 3).
- En la línea de producto D ninguna métrica, de las que se pudieron estudiar, parece estar relacionada (ver tabla 4).
- En la línea de producto O, pueden estar relacionadas las métricas GEvsC y MEvsC (ver tabla 5).

Cuando se estudió la correlación en el total de assets se observaron valores de p significativos en las métricas ASvsC con DEvsC (p =0.0598, n=27, Rs = 0.3692) y en AvsC con DEPvsC (p=0.0187, n=27.

Rs = -0.4613). Es de notar que en el segundo caso la relación, si existe es negativa, es decir, si AvsC y DEPvsC miden la complejidad, cuanto más contribuya a la complejidad una menos la otra.

Como podemos observar, analizando lo coeficientes de correlación, en general, las métricas propuestas son independientes, en las tres líneas de producto.

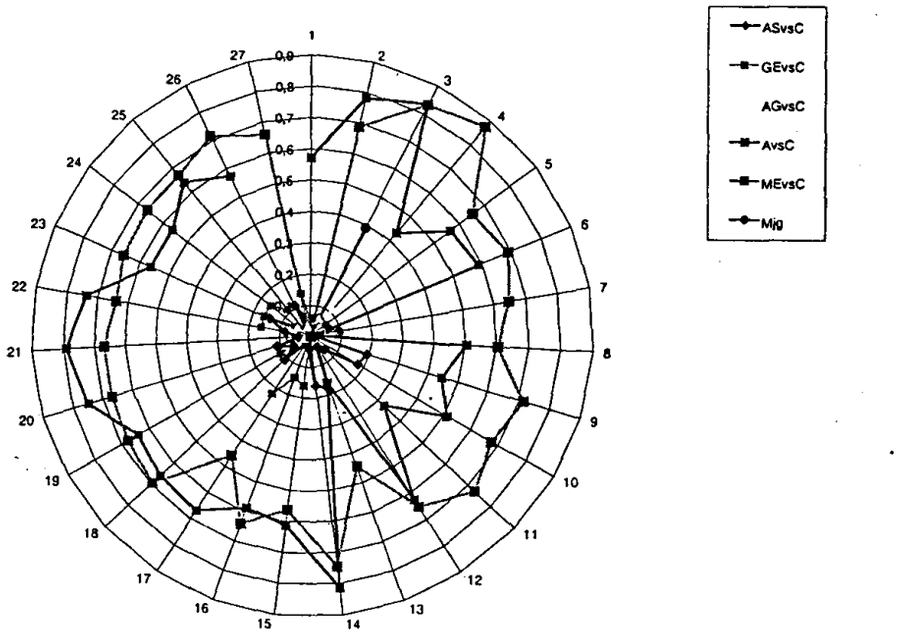

Figura 2: Todas las Líneas de producto

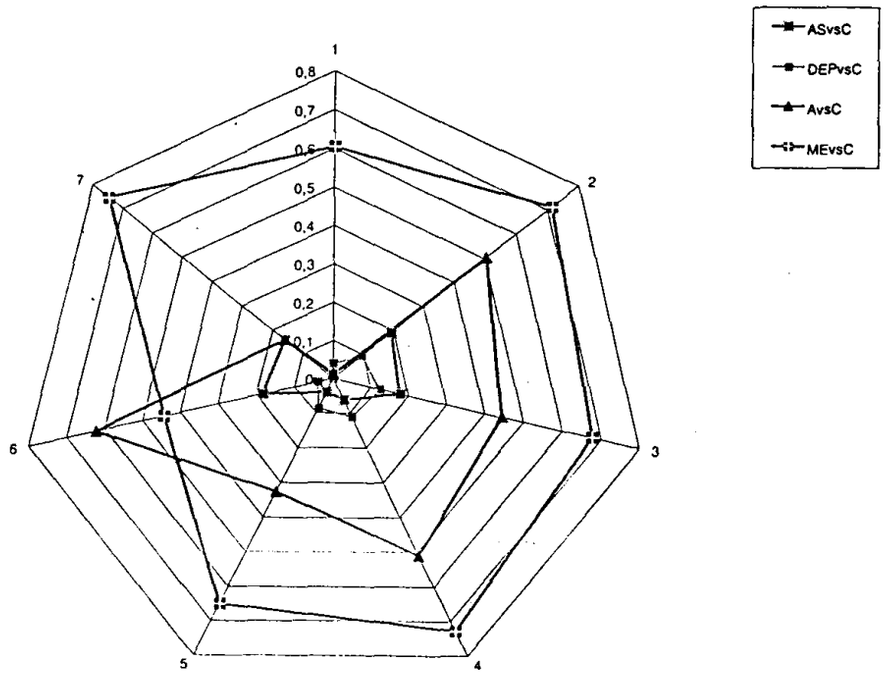

Figura 3. Linea de Producto de Discapacitados

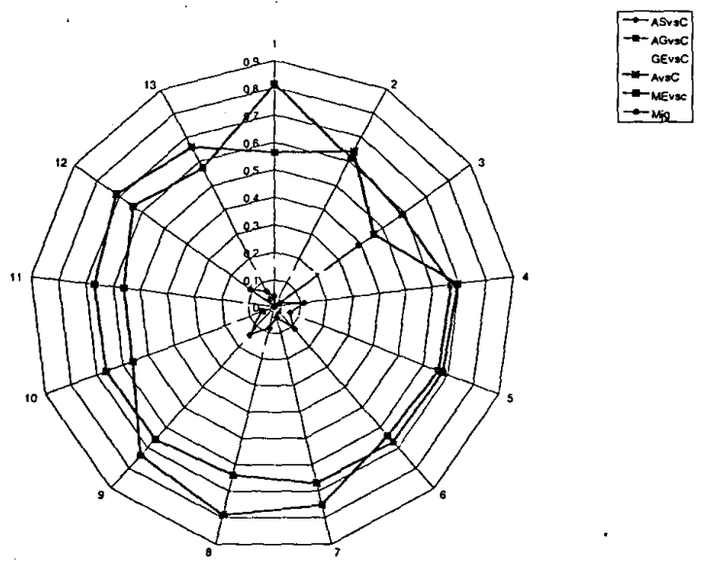

Figura 4. Línea de producto de Ópticas

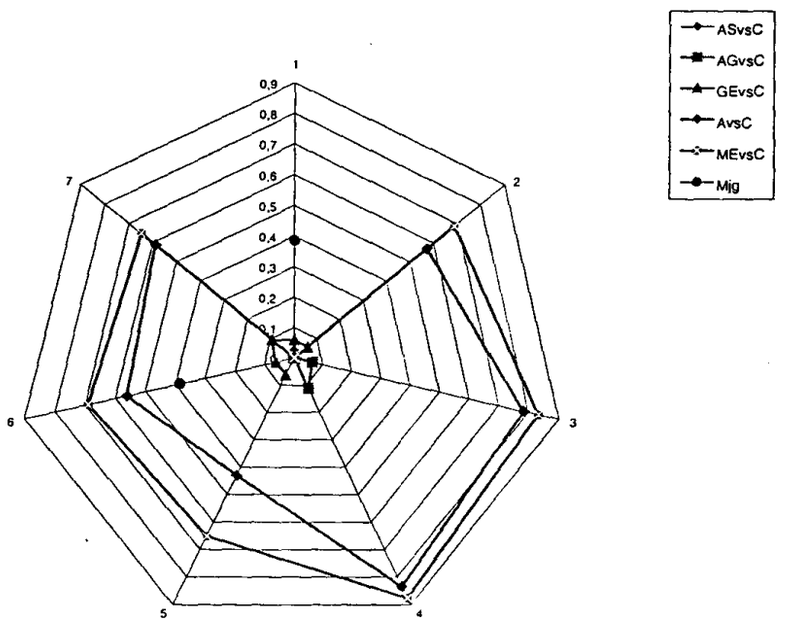

Figura 5. Línea de producto de Imagen

4. Conclusiones

En este trabajo se han propuesto un conjunto de métricas que miden la complejidad de los diagramas de clase expresados en UML. La ventaja principal de estas métricas con respecto a la mayoría de las existentes es que pueden aplicarse al diseño de un producto de software OO, es decir en las etapas tempranas del ciclo de vida, proporcionándole a los diseñadores información útil para mejorar las características internas del producto antes de su implementación.

Esta propuesta no debe considerarse una propuesta final, es solo un punto de partida. La definición de estas métricas, como cualquier otras, forma parte de un proceso de realimentación para poder refinarlas y ajustarlas a las necesidades.

Como en otros aspectos de la Ingeniería del Software, proponer técnicas y métricas no basta, es necesario validarlas de forma empírica y teórica, para justificar su utilidad. La validación es un tema crítico para el éxito de la medición de cualquier producto software (Kitchenham, et al., 1995).

En (Genero et. al, 2000a Genero, et al., 2000b) hemos presentado la validación teórica de alguna de estas métricas, siguiendo el marco formal de Briand (1996b). En el futuro validaremos esta métricas siguiendo el marco formal propuesto por Zuse (1998).

Estamos de acuerdo con varios autores como Fenton y Pflegeer (1997), Kitchenham et al. (1995), Schneidewind (1992), quienes sostienen que es necesario validar las métricas empíricamente para demostrar que las métricas funcionan "realmente" en la práctica.

En este trabajo hemos aplicado las métricas propuestas a diagramas de clases de tres líneas de producción, tomados del repositorio GIRO. El objetivo de la validación empírica realizada es:

1) Analizar si las métricas de complejidad en estudio se comportan de forma distinta en las 3 líneas de producto.
2) Analizar la independencia de la métricas.

Si bien el tamaño de la muestra sobre la que se realizó la validación empírica no es muy grande, consideramos que los resultados obtenidos son de utilidad, y constituyen un buen punto de partida. Aunque somos conscientes que es necesario seguir validando estas métricas con casos reales, para llegar a cumplir con nuestro objetivo final que es:

1) evaluar estas métricas como predictoras del esfuerzo del esfuerzo de mantenimiento, y luego determinar si pueden usarse como indicadores de la calidad en etapas tempranas del ciclo de vida.
2) proporcionarles a los diseñadores "valores límites" que puedan ser de utilidad a la hora de tomar decisiones en sus tareas de diseño de sistemas.

También es cierto que UML ha surgido hace relativamente poco tiempo, y por lo tanto será posible realizar mayor validación empírica. cuando haya disponible más datos sobre proyectos desarrollados utilizando este estándar.

Nuestro objetivo inmediato es seguir investigando en el tema de las métricas OO para proponer otras que consideren otros aspectos que influyen en la calidad del software OO, como la cohesión, el acoplamiento, reutilización, etc.

Igualmente es objetivo de nuestro grupo de investigación el desarrollo de una herramienta que permita calcular y analizar los valores de las métricas de forma automática.

Agradecimiento

Este trabajo de investigación es parte del subproyecto TIC97-0593-C05-05, del proyecto MENHIR financiado por el proyecto CICYT TIC97-0593-C01, y también parte del proyecto MANTICA, parcialmente financiado por la CICYT y la Unión Europea (IFD97-0168).

Referencias

- Booch, G., Rumbaugh J. y Jacobson, I. (1998). *The Unified Modeling Language User Guide.* Addison-Wesley.

- Briand, L., Differding, C. y Rombach, D. (1996a). Practical Guidelines for Measurement-Based Process Improvement. *Software Process-Improvement and Practice* 2, 253-280.
- Briand, L., Morasca, S. y Basili, V. (1996b). Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering*, 22 (6), 68-86.
- Brito e Abreu, F. y Carapuça, R. (1994). Candidate Metrics for Object Oriented Software within a Taxonomy Framework. *Journal of Systems and Software*, 26(1), North-Holland, Elsevier Science.
- Chidamber, S. y Kemerer, C. (1994). A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*. 20(6), 476-493.
- Fenton, N. (1994). Software Measurement: A Necessary Scientific Basis. *IEEE Transactions on Software Engineering*, 20(3), 199-206.
- Fenton E. y Pfleeger, S. (1997). *Software Metrics. A Rigurous and Practical Approach.* International Thompson Computer Press.
- García F.J., Marqués, J.M. y Maudes, J.M. (1997). Mecano: Una propuesta de Componente Software Reutilizable. *Actas de las II Jornadas de Ingeniería de Software* (Donostia- S. Sebastián, España, 3-5 septiembre de 1997), 232-244.
- García F.J., Marqués, J.M., Laguna M.A. y Maudes, J.M. (1998). Estructuras Complejas de reutilización: Definición de Mecano Estático. *Actas de las II Jornadas de trabajo MENHIR.* Editor José A. Carsí (Valencia, 19-20 de Febrero de 1998), 135-141. 1998.
- Genero, M., Manso, Mª E., Piattini,M. y García, F. (1999). Assessing the Quality and the Complexity of OMT Models. *2nd European Software Measurement Conference - FESMA 99*, Amsterdam, The Netherlands. 99-109.
- Genero, M., Piattini, M. y Calero, C. (2000a). Métricas para diagramas de clases UML. *Revista Novática, 143*, Enero-Febrero.
- Genero, M., Piattini, M. y Calero, C. (2000b). Métricas para jerarquías de agregación en diagramas de clases UML. *IDEAS 2000.* 5-7 Abril, Cancún, México.
- Henderson-Sellers, B. (1996). *Object-Oriented Metrics - Measures of complexity.* Prentice-Hall, Upper Saddle River, New Jersey.
- ISO/IEC 9126-1.2 (1999). Information technology- Software product quality – Part 1: Quality model.
- Kitchenham, B., Pflegger, S. y Fenton, N. 1995. Towards a Framework for Software Measurement Validation. *IEEE Transactions of Software Engineering*, 21(12), 929-943.
- Lethbridge, T. (1998). Metrics For Concept-Oriented Knowledge bases. *International Journal of Software Engineering and Knowledge Engineering* 8(2), 161-188.
- Li, H. y Cheng, W. (1987). An empirical study of software metrics. *IEEE Trans. on Software Engineering*, 13 (6): 679-708.
- Manso E, García F.J., Rodríguez J.J, Laguna M.A. (1999). Modelo de cualificación de asstes del repositorio GIRO. *Actas de las III Jornadas de trabajo MENHIR.* Editor Moros B. & Sáez J. (Murcia, 13-14 de Noviembre de 1998), 109-114.
- Pfleeger, S. (1997). Assessing Software Measurement. *IEEE Software.* March/April, 25-26.
- Schneidewind, N. 1992. Methodology For Validating Software Metrics. *IEEE Transactions of Software Engineering*, 18(5), 410-422.
- Zultner, R. (1992). The Deming Way: Total Quality Management for Software. *Proceedings of Total Quality Management for Software Conference*, April, Washington, DC, April.
- Zuse, H. (1998). A Framework of Software Measurement. Berlin, Walter de Gruyter.