Métricas para Modelos UML

Marcela Genero, Mario Piattini Velthuis, José Antonio Cruz-Lemus y Luis Reynoso
Grupo ALARCOS, Depto. de Informática, Universidad de Castilla-La Mancha

<{marcela.genero, mario.piattini, joseantonio.cruz}@uclm.es>, <lreynoso@uncoma.edu.ar>

Nota del Editor de *Novática*: la versión inglesa de este artículo fue publicada en **UPGRADE** (Vol. V, issue no. 2, April 2004), no habiendo sido incluido en la monografía del núm. 168 de ***Novática*** (marzo-abril 2004) por razones de espacio.

Resumen: *dado el importante papel que los modelos obtenidos en las etapas iniciales juegan en el desarrollo de sistemas orientados a objetos, en los últimos años se ha prestado una atención especial a la calidad de estos modelos. Aún cuando se cree que la aparición de UML* (Unified Modeling Language) *contribuye a construir sistemas software de mejor calidad, es necesario contar con instrumentos de medición cuantitativos y objetivos que nos permitan alcanzar este fin. Además, el hecho de que el uso de métricas para modelos UML puede ayudar a los diseñadores a tomar mejores decisiones está adquiriendo relevancia en el campo de la medición de software. Por esta razón, en los últimos años se ha incrementado la demanda de métricas para modelos UML. Por tanto, el principal objetivo de este trabajo es presentar un amplio panorama de las propuestas más relevantes relacionadas con métricas para modelos UML a nivel conceptual. Así, los investigadores y los profesionales pueden tener una nueva perspectiva sobre el trabajo ya hecho y aquel que todavía ha de hacerse para medir características de calidad de modelos UML. Este trabajo también ayudará a conseguir una visión bastante completa sobre la dirección que está tomando la medición OO.*

Palabras clave: *calidad de modelos, características, métricas orientadas a objetos, modelos conceptuales, modelos UML, validación empírica, validación teórica.*

Autores

Marcela Genero es Profesora Asociada en el Depto. de Informática de la Universidad de Castilla-La Mancha, Ciudad Real, España. Ingeniera en Informática por el Departamento de Informática de la Universidad del Sur, Argentina en 1989 y Doctora en Informática por la Universidad de Castilla-La Mancha, Ciudad Real, España. Sus principales líneas de investigación son: diseño avanzado de bases de datos, métricas de software, calidad en modelos de datos conceptuales, calidad en bases de datos. Ha publicado diversos artículos en prestigiosas conferencias y revistas CAiSE, E/R, OOIS, METRICS, ISESE, SEKE, Journal of Systems and Software, International Journal of Software Engineering and Knowledge Engineering, Information and Software Technology, Software Quality Journal, etc. Además, es coeditora del libro "Information and database quality", 2002, Kluwer Academic Publishers, USA y "Metrics for Software Conceptual Models", 2004, Imperial College Press, UK.

Mario Piattini Velthuis es Ingeniero y Doctor en Informática por la Universidad Politécnica de Madrid. Es también Certified Information System Auditor Manager (CISA) por la ISACA (Information System Audit and Control Association). Catedrático en el Depto. de Informática de la Universidad de Castilla-La Mancha, en Ciudad Real, España. Autor de diversos libros y artículos sobre bases de datos, Ingeniería del Software y sistemas de información, dirige el grupo de investigación ALARCOS del Depto. de Informática de la Universidad de Castilla-La Mancha, en Ciudad Real. Sus líneas de investigación son: diseño avanzado de bases de datos, calidad de bases de datos, métricas de software, mantenimiento de software y seguridad en sistemas de información. Ha coeditado diversos libros: "Advanced Databases: Technology and Design", 2000, Artech House, UK; "Auditing Information Systems" Idea Group Publishing, 2000, USA; "Information and database quality", 2002, Kluwer Academic Publishers, USA y "Metrics for Software Conceptual Models", 2004, Imperial College Press, UK. Es coordinador de la sección técnica "Bases de datos" de ***Novática***.

José Antonio Cruz-Lemus es Profesor Asociado en el Depto. de Informática de la Universidad de Castilla - La Mancha, España. Ingeniero en Informática por la Universidad de Castilla-La Mancha en 2003 y estudiante de doctorado en la misma universidad. Sus líneas de investigación son: metamodelado usando UML, métricas para modelos UML, etc.

Luis Reynoso es Profesor Asociado en la Universidad Nacional del Comahue, Neuquen, Argentina. Ingeniero en Informática por la Universidad Nacional del Sur, Argentina en 1993. Master en Informática en la misma universidad en 2003. Alumno del Instituto Internacional de Tecnología del Software, uno de los centros de Investigación y Formación de la Universidad de las Naciones Unidas, donde investigó en un proyecto sobre diseño de patrones orientados a objetos. Ha publicado artículos en diversas conferencias internacionales. Sus líneas de investigación se centran en métricas orientadas a objetos y la combinación de métodos formales e informales aplicados a la ingeniería del software.

1. Introducción

En los últimos años, los desarrolladores de software han prestado una atención especial a garantizar las características de calidad de los sistemas orientados a objetos (OO) desde las etapas iniciales de su ciclo de vida, centrándose especialmente en la calidad de los modelos conceptuales. A pesar de que la fase de modelado conceptual representa solamente una pequeña porción del esfuerzo global del desarrollo del sistema, su impacto sobre el sistema que finalmente se implementa es probablemente mayor que el de cualquier otra fase.

Recientemente, algunos paradigmas como el Desarrollo Dirigido por Modelos (MDD, *Model-Driven Development* [1]) y la Arquitectura Dirigida por Modelos (MDA, *Model-Driven Architecture* [27]) consideran que los modelos conceptuales son la espina dorsal del desarrollo de los sistemas OO, recalcando la importancia de construir "buenos" modelos conceptuales.

Si bien la aparición de UML (*Unified Modeling Language*) como lenguaje de modelado ha supuesto un gran avance en pro de construir modelos de mayor calidad, es cierto que un lenguaje de modelado sólo proporciona una sintaxis y una semántica para trabajar con ella, pero no indica si un modelo es 'bueno' o no.

Naturalmente, incluso cuando se domina un lenguaje, esto no garantiza que los modelos que se produzcan vayan a ser buenos. Es lo mismo que escribir una historia en lenguaje natural, el lenguaje es simplemente una herramienta que el autor debe dominar, pero sigue estando en manos del autor conseguir que la historia escrita sea buena.

Tradicionalmente, la calidad en el modelado conceptual ha sido un área poco comprendida. En la práctica, la evaluación de la calidad de los modelos conceptuales se hace de manera *ad hoc*, si es que se hace. No existen unas guías comúnmente aceptadas para evaluar la calidad de los modelos conceptuales, y hay muy poco consenso, incluso entre expertos, sobre qué hace que un modelo sea 'bueno'. Como consecuencia, la calidad de los modelos conceptuales que se producen en la práctica depende casi por completo de la competencia del modelador.

Los modeladores expertos saben intuitivamente qué hace que un modelo sea bueno, pero este conocimiento generalmen-

te sólo se adquiere a través de la experiencia. Sin embargo, para que el modelado conceptual pase de ser una "actividad artesanal" a convertirse en una disciplina ingenieril, deben explicitarse las características de calidad de los modelos conceptuales.

Se han propuesto varios marcos de trabajo interesantes [21][19][26][31], que han contribuido a una mejor comprensión del concepto de calidad en el modelado conceptual. Aunque estos marcos carecen de medidas cuantitativas y objetivas que permitan reducir la subjetividad y los prejuicios en el proceso de evaluación de la calidad.

Se reconoce ampliamente que sólo definir los criterios de calidad no es suficiente para asegurar la calidad en la práctica, porque generalmente se hacen diferentes interpretaciones del mismo concepto. De acuerdo con la literatura de Gestión de Calidad Total (TQM, *Total Quality Management*), se necesitan criterios medibles para evaluar la calidad, para así evitar "argumentos de estilo". Es en este contexto donde la medición de software juega un papel importante. Una métrica es una forma de medir una característica de calidad de manera consistente y objetiva.

Es necesario, por tanto, contar con métricas que permitan evaluar características que aseguren la calidad de los modelos conceptuales (específicamente en nuestro caso métricas para modelos UML) y contribuir de esta manera a desarrollar sistemas OO de calidad, eliminando atributos de diseño no correctos, disminuyendo la complejidad no deseada desde las etapas iniciales del ciclo de desarrollo, etc. Esto llevaría a una reducción considerable del trabajo extra necesario durante y después de la implementación.

Además debemos tener en cuenta que en muchos casos los modelos UML deben ser complementados con especificaciones OCL (*Object Constraint Language*), que permiten expresar ciertas restricciones imposibles de expresar sólo con las notaciones que aportan los distintos diagramas.

El objetivo principal de este trabajo es presentar una visión general de la literatura existente sobre métricas que pueden aplicarse a la medición de atributos de calidad de diagramas estructurales de UML (diagramas de clases), diagramas de comportamiento de UML (diagramas de caso de uso, diagramas de estados) y expresiones OCL.

Varios expertos han puesto especial énfasis en algunas cuestiones que deben tenerse en cuenta a la hora de definir métricas de software. Entre ellas:

- Las métricas deben definirse persiguiendo metas claras.
- Las métricas deben validarse teóricamente, contestando a la pregunta "*¿mide la métrica el atributo que pretende medir?*".
- Las métricas deben validarse empíricamente, contestando a la pregunta "*¿es útil la métrica en la práctica?*".
- El cálculo de las métricas debe ser fácil y aún mejor si pueden calcularse de manera automática a través de una herramienta.

Consideraremos estas sugerencias cuando presentemos cada una de las propuestas de métricas.

El resto de este trabajo se organiza así: las secciones 2 a la 4 esbozan las diferentes propuestas de métricas para diagramas de UML: diagramas de casos de uso, diagramas de clases y diagramas de estados, respectivamente. Las métricas para expresiones OCL se presentan en la sección 5. Finalmente, la última sección presenta algunas conclusiones y tendencias futuras y emergentes en el área de las métricas para modelos UML.

2. Métricas para diagramas de casos de uso de UML

En realidad, UML sólo se centra en la notación de los diagramas de los casos de uso. Como varios autores comentan, los diagramas de casos de uso deben entenderse únicamente como una tabla de contenido de los propios casos de uso, no como una alternativa para su especificación textual. En los diagramas de casos de uso sólo se muestran el nombre de los casos de uso, los actores que participan y algunas relaciones entre casos de uso. La esencia de los casos de uso, es decir, su secuencia de interacciones actor-sistema, no puede derivarse de ninguna manera a partir de los diagramas de casos de uso. Así pues, es necesario complementar los diagramas de casos de uso con la especificación textual de los casos de uso.

Existen pocas propuestas de métricas específicas para diagramas de casos de uso, tales como [23] y [30]. También existen otras propuestas específicas para casos de uso, como por ejemplo las propuestas de Feldt [14], Henderson-Sellers et al. [18] y Bernárdez et al. [4]. Un estudio más profundo sobre métricas para casos de uso y diagramas de casos de uso puede encontrarse en [15].

Como nuestra intención es mostrar las métricas existentes para diagramas de casos de uso, a continuación comentaremos brevemente las métricas propuestas por Marchesi [23] y Saeki [30].

Marchesi [23] propuso un conjunto de métricas de complejidad para diagramas de casos de uso consistente en el número de casos de uso (N_{CU}), el número de actores (N_a) y el número de relaciones de inclusión y extensión. Basándose en estas métricas también definió un indicador global de la complejidad del sistema.

En Saeki [30], se definen un conjunto de métricas para diagramas de casos de uso, con el fin de obtener un indicador de modificabilidad. La idea básica de estas métricas que es que si se necesita cambiar un caso de uso, probablemente habrá que cambiar otros casos de uso: aquellos que tengan una relación con el caso de uso que se cambió originalmente. En resumen, se consideran las relaciones de inclusión y extensión y las relaciones de control y de dependencia de datos.

La intuición aconseja que, cuantas más relaciones haya en el modelo, más difícil será hacer cualquier cambio.

Otro factor que influye en la modificabilidad de los casos de uso es el tipo de caso de uso. Simplificando la idea, si un caso tiene varios objetivos (tipos para Saeki), es más susceptible de cambiar que si tiene sólo un objetivo. Para aproximar la modificabilidad, se definen las métricas *NOD* (Número de Dependencias) y *NUCT* (Número de Tipos de Casos de Uso). El objetivo alcanzado por el autor consistió en un indicador ($0 \leq MODIFIABILITY \leq 1$) que revelaba el gra-

Métricas	Definición
WMC (*Weighted Methods per Class*)	La métrica Métodos Ponderados por Clase se define como: $WMC = \sum_{i=1}^{n} c_i$ Donde c_1, ..., c_n representan la complejidad de los métodos de una clase con métodos M_1, ...,M_n. Si se considera que todos los métodos tienen complejidad 1, WMC = n, el número de métodos.
DIT (*Depth of Inheritance Tree*)	La Profundidad de Herencia de una clase es la métrica DIT de la clase. En los casos con herencia múltiple, DIT será la longitud máxima desde el nodo a la raíz del árbol.
NOC (*Number of Children*)	El Número de Hijos es el número de subclases inmediatas subordinadas a una clase en la jerarquía de clases..

Tabla 1. Métricas CK [12].

Métricas	Definición
Nassoc	Número total de asociaciones.
Nagg	Número total de relaciones de agregación dentro de un diagrama de clases (cada par todo-parte de una relación de agregación).
Ndep	Número total de relaciones de dependencia.
Ngen	Número total de relaciones de generalización dentro de un diagrama de clases (cada par padre-hijo de una relación de generalización).
NaggH	Número total de jerarquías de agregación (estructuras todo-parte) dentro de un diagrama de clases.
NgenH	Número total de jerarquías de generalización dentro de un diagrama de clases.
MaxDIT	El valor máximo de la métrica DIT (*Depth of Inheritance Tree*. Profundidad del Árbol de Herencia) calculada para cada clase del diagrama de clases. El valor de DIT para una clase dentro de una jerarquía de generalización es el camino más largo desde la clase hasta la raíz de la jerarquía.
MaxHAgg	El valor máximo de la métrica HAgg calculada para cada clase del diagrama de clases. El valor de HAgg para una clase dentro de una jerarquía de agregación es el camino más largo desde la clase hasta las hojas.

Tabla 2. Métricas para complejidad estructural de diagramas de clases UML.

do de modificabilidad de un modelo de casos de uso.

No conocemos que haya ninguna prueba de validación teórica o empírica de estas dos propuestas. Además, no existe soporte automático para el cálculo de las métricas.

3. Métricas para diagramas de clases de UML

La idea principal de esta sección es mostrar un resumen de las propuestas más relevantes de métricas[1] que pueden aplicarse a los diagramas de clases de UML a nivel conceptual, estudiando sus puntos fuertes y débiles. La mayoría de las propuestas de métricas que consideraremos y comentaremos a continuación, no se definieron originalmente para medir diagramas de clases de UML, aunque pueden adaptarse para este propósito:

- Chidamber y Kemerer [12]. Estas métricas, también llamadas métricas CK, se definieron a nivel de clase y su propósito es medir la complejidad del diseño en relación a su impacto en los atributos externos de calidad tales como la mantenibilidad, la reusabilidad, etc. Esta propuesta es una de las que ha tenido mayor difusión y uso. Solamente tres de las seis métricas CK se pueden aplicar a un diagrama de clases de UML a nivel conceptual (**tabla 1**).
- Li y Henry [20]: Estas métricas, definidas a nivel de clase, miden diferentes atributos internos como el acoplamiento, la complejidad y el tamaño, y se utilizaron con éxito para predecir el esfuerzo del mantenimiento.
- Brito e Abreu y Carapuça [8]. Se definieron para medir el uso de mecanismos de diseño OO como la herencia, el ocultamiento de información, el acoplamiento y el polimorfismo y la relación existente con la calidad del software y la productividad del desarrollo. Pueden aplicarse a nivel de diagramas de clases.
- Lorenz y Kidd [22]: Se definieron a nivel de clase para medir características estáticas del diseño de software, como el uso de la herencia, la cantidad de responsabilidades de una clase, etc.
- Briand et al. [5]: Estas métricas se definieron a nivel de clase, y cuentan las interacciones entre clases. Su propósito es la medición del acoplamiento entre clases.
- Marchesi [23]. El objetivo de estas métricas es la medición de la complejidad del sistema, del equilibrio de las responsabilidades entre paquetes y clases, y de la cohesión y el acoplamiento entre entidades del sistema.
- Harrison et al. [17]: Propusieron la métrica Número de Asociaciones por clase como una métrica de acoplamiento entre clases.
- Genero et al. [16][15]: Definieron un conjunto de métricas para la complejidad estructural de diagramas de clase de UML debido al uso de relaciones de UML, tales como: asociaciones, generalizaciones, dependencias y agregaciones (**tabla 2**). Los autores suponen que estas métricas pueden ser buenos indicadores de las características de mantenibilidad de un diagrama de clases de UML.
- Bansiya et al [2][3]: Estas métricas se definieron a nivel de clase para evaluar propiedades de diseño como la encapsulamiento, el acoplamiento, la cohesión, la composición y la herencia.

La **tabla 3** refleja los estudios publicados acerca de la validación teórica y empírica de las propuestas de métricas mencionadas anteriormente. Además, la última columna indica si existen o no herramientas que permitan automatizar el cálculo de la métricas.

4. Métricas para diagramas de estados de UML

Las métricas para diagramas de estados de UML son escasas. Derr [13] definió el número de estados y el número de transiciones como métricas que miden la complejidad de los diagramas de estados OMT (*Object Modelling Technique*) — aunque también se pueden aplicar a UML).

Carbone et al. [11] propusieron dos métricas: totSta(c), que representa en numero total de estados para una clase y totAction(c) que cuenta el número total de acciones para una clase. Estas métricas son utilizadas junto a otras para diagramas de clases, diagramas de casos de uso, etc., para determinar la complejidad total de un sistema OO.

En cualquier caso, ambas propuestas no han pasado más allá de la definición.

Quizá el trabajo más completo es el propuesto por [25]. Con la hipótesis de que el tamaño y la complejidad estructural de los diagramas de estados de UML puede influenciar su facilidad de entendimiento (y por tanto su mantenibilidad), definieron un conjunto de métricas para la complejidad estructural y el tamaño de los diagramas de estados de UML.

Como métricas de tamaño definieron:

- NEntryA. Número total de acciones de entrada, es decir, las acciones que se realizan cada vez que se entra en un estado.

FUENTE	VALIDACIÓN				Herramienta
	EMPÍRICA		TEÓRICA		
	Experimentos	Casos de estudio	Enfoque basado en propiedades	Enfoque basado en teoría de la medida	
Chidamber y Kemerer [12]	SÍ	SÍ	SÍ	SÍ	SÍ
Li y Henry [20]		SÍ			SÍ
Brito e Abreu y Carapuça [8]		SÍ		SÍ	SÍ
Lorenz y Kidd [22]		SÍ			SÍ
Briand et al. [5]		SÍ	SÍ		SÍ
Marchesi [23]		SÍ			SÍ
Harrison et al. [17]		SÍ			
Bansiya et al. [2][3]		SÍ			SÍ
Genero et al. [16][15]	SÍ		SÍ	SÍ	SÍ

Tabla 3. Resumen de las propuestas de métricas para diagramas de clases de UML.

Técnica Cognitiva	Características comunes del grupo de conceptos OCL	Ejemplos de conceptos OCL relacionados con cada grupo
Tracing	Conceptos OCL que permiten que una expresión utilice propiedades que pertenezcan a otras clases o interfaces, diferentes al tipo de la instancia contextual.	Navegación y operación de colecciones, Mensajería, parámetros cuyos tipos son clasificadores definidos en el diagrama de clases, etc.
Chunking Grupo 1	Servicios OCL relacionados con el propio lenguaje.	Definiciones de variables a través de expresiones *let*, Expresiones condicionales *If*, variables iterativas predefinidas, literales, etc.
Chunking Grupo 2	Conceptos OCL relacionados con la instancia contextual y alguna de sus propiedades.	Referencia a atributos u operadores de la instancia contextual, valores postfijos de by @pre, variables definidas a través de restricciones <<definition>>, etc..

Tabla 4. Grupos definidos de conceptos OCL de acuerdo a técnicas cognitivas.

- NExitA. Número total de acciones de salida, es decir, las acciones que se realizan cada vez que se abandona un estado.
- NA. Número total de actividades (do/ activity).
- NSS. Número total de estados, considerando también los estados simples dentro de los estados compuestos.
- NCS. Número total de estados compuestos.
- NE. Número total de eventos.
- NG. Número total de condiciones de guarda.

Como métricas de complejidad estructural definieron:

- NT (*Number of Transitions*). Número total de transiciones, considerando las transiciones comunes (los estados origen y destino son diferentes), las transiciones inicial y final, las auto-transiciones (los estados origen y destino son el mismo) y las transiciones internas (transiciones dentro de un estado que responden a un evento pero sin dejar el estado).
- CC (McCabe's *Cyclomatic Complexity*, Número Ciclomático de McCabe) [22][2]. Se define como |NSS-NT|+2

La validez teórica de las métricas propuestas se demostró a través de la validación siguiendo el marco de trabajo de Briand et al., concluyendo que las métricas NA, NSS, NCS, NE, NEntryA, NExitA y NG son de tamaño; y las métricas NT y CC son de complejidad. Además, el uso del marco de trabajo DISTANCE [28] garantiza que las métricas pueden usarse como instrumentos de medición en escala de ratio.

Su hipótesis fue corroborada empíricamente en cierta medida a través de un experimento controlado y su réplica. Como resultado del trabajo de experimentación, concluyeron que las métricas NA, NSS, NG y NT parecen estar altamente correlacionadas con el tiempo de entendimiento de los diagramas de estados de UML. Sin embargo, estos hallazgos deben considerarse como preliminares.

5. Métricas para expresiones OCL

La única propuesta de métricas para expresiones OCL es la realizada por Reynoso et al. [29], quienes tomaron como hipótesis que las propiedades estructurales de una expresión OCL tienen un importante impacto en la complejidad cognitiva de los modeladores, ya que cuando los desarrolladores tratan de comprender una expresión OCL (considerada en este estudio como una abstracción mental simple: un '*chunk*') aplican técnicas cognitivas, como el *chunking* y el *tracing*. Estas técnicas se aplican concurrente y sinergéticamente en resolución de problemas y son parte del Modelo de Complejidad Cognitiva (*Cognitive Complexity Model*, CMM) definido por Cant et al. [9][10]. Este modelo proporciona una teoría cognitiva general de la complejidad del software relacionada con el impacto de la estructura sobre la comprensión. El *chunking* implica el reconocimiento de un conjunto de información de declaración y extracción, que se recuerda como un *chunk*, mientras que el *tracing* implica explorar, hacia adelante o hacia atrás, para identificar los *chunks*.

El *tracing* se ha observado como una actividad fundamental en la comprensión del software. Por esa razón, estos autores han definido métricas relacionadas con estas técnicas cognitivas, agrupándolas en tres grandes grupos de conceptos de OCL (**tabla 4**).

Aunque las expresiones OCL pueden añadirse a cualquier diagrama de UML, Reynoso et al. [29] han comenzado la definición de métricas para expresiones OCL que puedan usarse en un diagrama de clases de UML. Algunos ejemplos de métricas relacionadas con el *tracing* son (entre otras):

- Número de Relaciones Navegadas (*Number of Navigated Relationships*, NNR). Esta métrica cuenta el número total de relaciones que se navegan en una expresión. Si una relación se navega dos veces, por ejemplo usando propiedades diferentes de una clase o interfaz, esta relación se cuenta solamente una vez. Siempre que una *clase asociación* se navegue, consideraremos la asociación a la que la clase asociación va unida.
- Número de Atributos referidos a través de las Navegaciones (*Number of Attributes referred through Navigations*, NAN). Esta métrica cuenta el número total de atributos que son referidos a través de las navegaciones de una expresión.
- Número de Clases Navegadas (*Number of Navigated Classes*, NNC). Esta métrica cuenta el número total de clases, clases de asociación o interfaces a las que navega una expresión. Si una clase contiene una relación reflexiva y una expresión la navega, la clase se considerará sólo una vez en la métrica. Además, como una clase puede alcanzarse desde una clase/interfaz inicial navegando de forma diferente (i.e. siguiendo relaciones diferentes) debemos considerar esta situación como un caso especial: Si una clase se usa en dos (o más) navegaciones diferentes, la clase se cuenta sólo una vez.

Estas métricas también se validaron teóricamente usando el marco de Briand et al. [6][7]. Por ejemplo, las métricas NNR, NNC y NAN se validaron como métricas de acoplamiento basadas en la interacción. La intuición dice que las métricas relacionadas con el *tracing* podrían tener más influencia en la facilidad de entendimiento y la modificabilidad de las expresiones OCL. Pero este hecho debe comprobarse a través de estudios empíricos. No hay ninguna prueba de su validación empírica. Sin embargo, los autores planean realizar un experimento controlado para validar algunas de las métricas propuestas como indicadores de la facilidad de entendimiento y la modificabilidad de las expresiones OCL.

6. Conclusiones

Hemos querido presentar un resumen que cubre los trabajos más relevantes de la bibliografía sobre métricas para atributos de calidad de diagramas UML y expresiones OCL, a fin de proporcionar a profesionales una visión general de lo que se ha hecho en este campo y qué métricas están disponibles para poder ayudarles a tomar decisiones en las fases iniciales del desarrollo OO. Este trabajo también ayudará a los investigadores a conseguir una visión más completa de la dirección que está tomando la medición de modelos UML.

Pueden encontrarse más detalles de algunas

propuestas presentadas en este trabajo en las referencias bibliográficas donde las métricas fueron propuestas de manera original o bien en [15].

Como revela el presente estudio, los diagramas de clases UML han sido los más investigados desde el punto de vista de la medición. En menor media se han propuesto y validado métricas para diagramas de caso de uso de UML, diagramas de estados de UML y, casi ninguna para expresiones OCL. Otros modelos UML, que cubren aspectos dinámicos de los sistemas OO, como los diagramas de secuencia, los diagramas de actividad, etc., no se han tenido en cuenta, de momento, en el campo de la medición.

Aunque el número de métricas que se han propuesto para los diagramas de UML a nivel conceptual es bajo en comparación con el gran número de las definidas para código o diseño avanzado, creemos que en lugar de seguir definiendo nuevas métricas, se deben investigar las propiedades de las métricas existentes y su utilización a través de estudios empíricos y sus réplicas, cambiando así la dedicación de esfuerzos por parte de los investigadores.

Necesitamos entender mejor qué capturan realmente las métricas, si son realmente diferentes, y si son indicadores útiles de los atributos externos de calidad como la mantenibilidad, la productividad, etc. Una vez aclarados estos aspectos, aflorará la necesidad de nuevas métricas que debería ser guiada por los resultados de estos estudios.

En esta área, los diseñadores también demandan valores deseables para cada medida. Sin embargo, debemos ser conscientes de que asociar las calificaciones de bueno y malo con valores numéricos es la parte más difícil. Esto puede contribuir a que las métricas sean útiles para que los diseñadores de sistemas OO puedan tomar mejores decisiones en sus tareas de diseño, que es el objetivo más importante que debe perseguir cualquier propuesta de medición.

Como comentario final, podemos concluir que es evidente que el campo de las métricas para evaluar la calidad de los modelos UML debe madurar. Creemos que es necesaria más validación empírica, especialmente aplicando las métricas a modelos obtenidos en proyectos reales, para construir un cuerpo de conocimiento sólido sobre la utilidad de las métricas en la práctica.

Referencias

[1] C. Atkinson and T. Kühne. Model-Driven Development: A Metamodeling Foundation. IEEE Software, 20(5), pp. 36- 41, 2003.

[2] J. Bansiya and C. Davis. A Hierarchical Model for Object-Oriented Design Quality Assessment. IEEE Transactions on Software Engineering, 28(1), pp. 4-17, 2002.

[3] J. Bansiya, L. Etzkorn, C. Davis and W. Li. A Class Cohesion Metric For Object-Oriented Designs, The Journal of Object-Oriented Programming, 11(8), pp. 47-52, 1999.

[4] B. Bernárdez, A. Durán, and M. Genero. An empirical review of use cases metrics for requirements verification. Proceedings of the Software Measurement European Forum (SMEF'04), 2004.

[5] L. Briand, W. Devanbu and W. Melo. An investigation into coupling measures for C++, 19th International Conference on Software Engineering (ICSE 97), Boston, USA, pp. 412-421, 1997.

[6] L. Briand, S. Morasca and V. Basili. Property-Based Software Engineering Measurement. IEEE Transactions on Software Engineering, 22(1), pp. 68-86, 1996.

[7] L. Briand, S. Morasca and V. Basili. Defining and validating measures for object-based high level design. IEEE Transactions on Software Engineering, 25(5), pp. 722-743, 1999.

[8] F. Brito e Abreu and R. Carapuça. Object-Oriented Software Engineering: Measuring and controlling the development process, Proceedings of the 4th Int. Conference on Software Quality, McLean, VA, USA, pp. 3-5, 1994.

[9] S. Cant, B. Henderson-Sellers and R. Jeffery. Application of Cognitive Complexity Metrics to Object-Oriented Programs. Journal of Object-Oriented Programming, 7(4), pp. 52-63, 1994.

[10] S. Cant, R. Jeffery and B. Henderson-Seller. A Conceptual Model of Cognitive Complexity of Elements of the Programming Process. Information and Software Technology, 7, pp. 351-362, 1992.

[11] M. Carbone and G. Santucci. Fast&&Serious: a UML based metric for effort estimation. 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE´2002), pp. 35-44, 2002.

[12] S. Chidamber. and C. Kemerer. A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 20(6), pp. 476-493, 1994.

[13] K. Derr. Applying OMT. SIGS Books, Prentice Hall. New York, 1995.

[14] P. Feldt. Requirements metrics based on use cases. Master's thesis, Department of Communication Systems, Lund Institute of Technology, Lund University, Box 118, S-221 00 Lund, Sweden, 2000.

[15] M. Genero, M. Piattini and C. Calero (Eds.) Metrics For Software Conceptual Models. Imperial College Press, UK, 2004.

[16] M. Genero, M. Piattini and C. Calero. Early Measures for UML Class Diagrams. L'Objet, 6(4), Hermes Science Publications, 489-515, 2000.

[17] R. Harrison, S. Counsell and R. Nithi. Coupling Metrics for Object-Oriented Design. 5th International Software Metrics Symposium Metrics, pp. 150-156, 1998.

[18] B. Henderson-Sellers, D. Zowghi, T. Klemola and S. Parasuram. Sizing use cases: How to create a standard metrical approach. 8th Object–Oriented Information Systems, Lecture Notes in Computer Science, 2425, Springer–Verlag, pp. 409–421, 2002.

[19] J. Krogstie, O. Lindland and G. Sindre. Towards a Deeper Understanding of Quality in Requirements Engineering. 7th International Conference on Advanced Information Systems Engineering (CAISE), Jyvaskyla, Finland, June, 82-95, 1995.

[20] W. Li and S. Henry. Object-Oriented metrics that predict maintainability. Journal of Systems and Software, 23(2), pp. 111-122, 1993.

[21] O. Lindland, G. Sindre and A. Solvberg. Understanding Quality in Conceptual Modelling. IEEE Software, 11(2), pp. 42-49, 1994.

[22] M Lorenz and J. Kidd. Object-Oriented Software Metrics: A Practical Guide. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[23] M. Marchesi. OOA Metrics for the Unified Modeling Language. 2nd Euromicro Conference on Software Maintenance and Reengineering, pp. 67-73, 1998.

[24] T. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering.* 2(4), pp. 308-320, 1976.

[25] D. Miranda, M. Genero and M. Piattini. Empirical validation of metrics for UML statechart diagrams. Fifth International Conference on Enterprise Information Systems (ICEIS 03), 1, pp. 87-95, 2003.

[26] L. Moody, G. Shanks and P. Darke. Improving the Quality of Entity Relationship Models – Experience in Research and Practice. 17th International Conference on Conceptual Modelling (ER ´98), Singapore, pp. 255-276, 1998.

[27] Object Management Group. MDA-The OMG Model Driven Architecture. Available at <http://www.omg.org./mda/>, August 1st, 2002.

[28] G. Poels and G. Dedene. DISTANCE: A Framework for Software Measure Construction. Research Report DTEW9937, Dept. Applied Economics, Katholieke Universiteit Leuven, Belgium, p. 46, 1999.

[29] L. Reynoso, M. Genero and M. Piattini. Measuring OCL Expressions: An Approach Based on Cognitive Techniques. Chapter 5 in "Metrics for Software Conceptual Models" (Eds. Genero M., Piattini M. and Calero C.). Imperial College Press, UK. 2004.

[30] M. Saeki. Embedding Metrics into Information System Development Methods: An Application of Method Engineering Technique. Lecture Notes in Computer Science, 2681, pp. 374–389, 2003.

[31] R. Schuette and T. Rotthowe. The Guidelines of Modeling – An Approach to Enhance the Quality in Information Models. 17th International Conference on Conceptual Modelling (ER ´98), Singapore, pp. 240-254, 1998.

Notas

[1] Dada la enorme cantidad de métricas existentes que pueden aplicarse a los diagramas de clases UML a nivel conceptual, es imposible enumerarlas todas. Por tanto, a modo de ejemplo sólo mostraremos la definición de algunas. Más detalles sobre su definición y validación se puede encontrar en los artículos originales, donde se presentaron tales propuestas.

[2] Aunque el Número Ciclomático de McCabe se definió para calcular la complejidad de módulos simples y del sistema completo, se adaptó para medir la complejidad estructural de los diagramas de estados de UML.

/docs/ Gestión del conocimiento (Informe Capgemini/IE3)

Nº 170 julio-agosto 2004 año XXX

Un mundo de Agentes

novática
UPGRADE Revista de la Asociación de Técnicos de Informática

Novática, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de ATI (Asociación de Técnicos de Informática). *Novática* edita también *Upgrade*, revista digital de CEPIS (*Council of European Professional Informatics Societies*), en lengua inglesa, y es miembro fundador de UPENET (*UPGRADE European NETwork*)

<http://www.ati.es/novatica/>
<http://www.upgrade-cepis.org/>

ATI es miembro fundador de CEPIS (*Council of European Professional Informatics Societies*) y es representante de España en IFIP (*International Federation for Information Processing*); tiene un acuerdo de colaboración con ACM (*Association for Computing Machinery*), así como acuerdos de vinculación o colaboración con AdaSpain, AI2 y ASTIC.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid
Padilla 66, 3º, dcha., 28006 Madrid
Tlfn.914029391. fax.913093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tlfn./fax 963330392 <secreval@ati.es>
Administración y Redacción ATI Cataluña
Via Laietana 41, 1º, 1ª, 08003 Barcelona
Tlfn.934125235, fax 934127713 <secregen@ati.es>
Redacción ATI Andalucía
Isaac Newton, s/n, Ed. Sadiel,
Isla Cartuja 41092 Sevilla, Tlfn./fax 954460779 <secreand@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Asturias-Cantabria <gp-astucant@ati.es>
Redacción ATI Castilla-La Mancha <gp-clmancha@ati.es>
Redacción ATI Galicia
Recinto Ferial s/n, 36540 Silleda (Pontevedra)
Tlfn.986581413, fax 986580162 <secregal@ati.es>
Suscripción y Ventas
<http://www.ati.es/novatica/interes.html>, o en ATI Cataluña o ATI Madrid
Publicidad
Padilla 66, 3º, dcha., 28006 Madrid
Tlnf.914029391. fax.913093685 <novatica.publicidad@ati.es>
Imprenta
9 Impressió S.A., Juan de Austria 66, 08005 Barcelona
Depósito legal: B-15.154-1975 - ISSN 0211-2124- CODEN NOVAEC
Portada: Antonio Crespo Foix / © ATI 2004
Diseño: Fernando Agresta / © ATI 2004

Nº 170, julio-agosto 2004, año XXX

sumario

Monografía del próximo número: "Tecnologías de Proceso Software"