Definición de métricas para la complejidad de expresiones OCL de forma metodológica.

Luis Reynoso[1], Marcela Genero[2], Mario Piattini[2]
[1] Universidad Nacional del Comahue
Departamento de Ciencias de Computación, Facultad de Economía y Administración
Buenos Aires 1400, Neuquén, Argentina
Teléfono:+ 54 299 4490312 Fax: +54 299 4490313
lreynoso@uncoma.edu.ar , lreynoso@proyectos.inf-cr.uclm.es,
[2] Universidad de Castilla La Mancha
Grupo de Investigación Alarcos
Paseo de la Universidad 4, 13071, Ciudad Real, España
Teléfono: +926 295 300 Fax: +926 295 354
{Marcela.Genero, Mario.Piattini}@uclm.es

RESUMEN

Dado que los diagramas de clases constituyen "la columna vertebral" del desarrollo de software orientado a objetos (OO), han surgido muchas propuestas de métricas para medir atributos internos de su calidad como la complejidad estructural, el acoplamiento, el tamaño, etc. Pero ninguna de las propuestas existentes considera la complejidad añadida a los diagramas de clases UML al incorporarles expresiones escritas en el "Object Constraint Language" (OCL). Es bien sabido que el lenguaje OCL realmente enriquece a los diagramas de clases ya que los complementa a través de expresiones que especifican propiedades semánticas del modelo, mejorando la precisión del sistema, su documentación, y su comprensibilidad en etapas iniciales del desarrollo. Esto es lo que nos llevó a definir un conjunto de métricas para la complejidad estructural de las expresiones OCL considerando sólo aquellos elementos de OCL que se ven implicados en técnicas de "tracing". Consideramos que las técnicas de "tracing" afectan en gran medida a la complejidad cognitiva y a la comprensibilidad de las expresiones OCL, lo que afectará al mantenimiento de los diagramas de clases UML.

El principal objetivo de este artículo es presentar el estado de trabajo de investigación que se está desarrollando como parte de una tesis doctoral, poniendo especial énfasis en el proceso metodológico utilizado para la obtención de métricas válidas.

Palabras clave

Métricas OO, diagramas de clases, UML, complejidad estructural, complejidad cognitiva, comprensibilidad, mantenibilidad, validación teórica, validación empírica.

1 INTRODUCCIÓN

En la producción de software como en todo proceso de ingeniería, es ampliamente conocido que el uso de métricas en las fases iniciales de ciclo de vida ayuda a diseñadores a tomar mejores decisiones y a predecir atributos de calidad externos, tal como la comprensibilidad y la facilidad de mantenimiento [10]. En estas etapas iniciales uno de los artefactos claves es el diagrama de clases, ya que en ellos se basa todo el trabajo de diseño e implementación posterior. Por ello es necesario prestar especial atención a la calidad de los diagramas de clases. En la literatura actual existen numerosas métricas que pueden aplicarse a diagramas de clases UML en una etapa de alto nivel [9], [11], [2], [8], [13]. La mayoría de estos trabajos se centran en la medición de atributos internos de la calidad como la complejidad estructural, el acoplamiento, el tamaño, etc. Pero ninguna

de las propuestas de métricas existentes considera la complejidad añadida a los diagramas de clases UML al incorporarle expresiones escritas en OCL. Sin lugar a dudas la aparición del lenguaje OCL, definido por OMG, ha resultado fundamental en el desarrollo de software OO utilizando UML, ya que permite obtener un modelamiento preciso con UML, proveyendo al modelador de una notación expresiva para capturar las propiedades esenciales del sistema [14]. OCL realmente enriquece los diagramas UML ya que los complementa con expresiones que especifican propiedades semánticas del modelo [12], mejoran la precisión del sistema, su documentación [17], y su comprensibilidad en etapas iniciales del desarrollo.

Según nuestro conocimiento, el hecho de que una clase tenga asociadas expresiones OCL incrementa su complejidad estructural y hace que un diagrama de clases UML sea más difícil de entender y por consiguiente de mantener.

Esto nos llevó a pensar en la necesidad de contar con métricas para medir la complejidad estructural de expresiones OCL.

Briand et al. [4] han definido una base teórica para el desarrollo de modelos cuantitativos que relacionan la complejidad estructural y atributos de calidad externos. Asumimos en este trabajo una representación similar para expresiones OCL. Implementamos la relación entre la complejidad estructural por un lado, y los atributos de calidad externos por el otro (ver figura 1). Nuestra hipótesis es que las propiedades estructurales (tales como complejidad estructural) de una expresión OCL tienen un impacto en su complejidad cognitiva. El concepto de complejidad cognitiva significa la carga mental de las personas que tienen que tratar con artefactos (por ej. desarrolladores, personas que realizan verificación o mantenimiento). Una alta complejidad cognitiva conduce a que una expresión reduzca su comprensibilidad y esto conduce a cualidades externas indeseables, tal como una mantenibilidad menor.

Suponemos que las propiedades estructurales de expresiones OCL, como la complejidad estructural, tienen un impacto en la complejidad cognitiva de los modeladores, debido a que al querer comprender expresiones OCL se emplean técnicas cognitivas, tales como "chunking" y "tracing"[1] [6].

Por todo lo dicho, el siguiente artículo pretende mostrar dos objetivos alcanzados:

1. Definir un conjunto de métricas para la complejidad estructural de expresiones OCL, considerando sólo aquellos elementos de OCL especificados en su metamodelo [16], que afectan al "tracing".
2. Presentar el método que comenzamos a utilizar en nuestra investigación para la definición de métricas, con el fin de obtener medidas que sean válidas tanto desde un punto de vista teórico como práctico.

El resto del artículo está organizado de la siguiente manera: en la sección 2 se resume cada una de las etapas de un método de definición de métricas. En la sección 3 se resume la definición de las métricas y su validación teórica. Finalmente, se presentan las conclusiones y aquellas líneas de trabajo que quedan abiertas para una futura investigación.

2 UN MÉTODO PARA LA DEFINICIÓN DE MÉTRICAS

Para obtener métricas válidas es necesario definirlas de manera disciplinada. Por ello, en esta sección describimos un método basado en las propuestas de Calero et al. [5] y Cantone y Donzelli [7]. Este método consta de las siguientes cinco etapas:

- *Identificación*: En esta etapa se definen los objetivos que se persiguen a la hora de crear la métrica y se plantean las hipótesis de cómo se llevará a cabo la medición. Sobre

[1] La técnica de chunking, una capacidad de la memoria a corto plazo, involucra el reconocimiento de grupos de declaraciones (no necesariamente secuenciales) y extraer de ellas información que es recordada en una única abstracción mental: un chunk.

Las técnicas de tracing involucran el barrido de información, en diferentes direcciones, con el objetivo de identificar chunks relevantes [39] para resolver determinadas dependencias de conceptos.

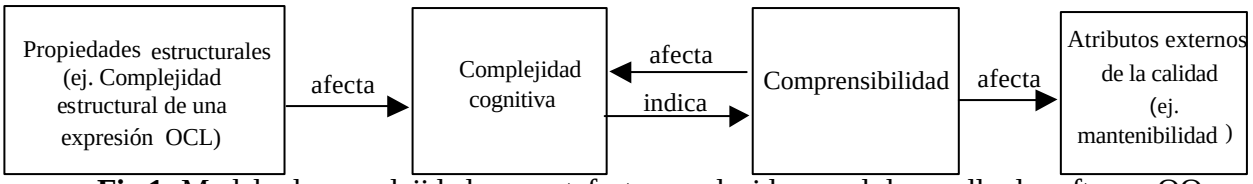

Fig 1: Modelo de complejidad para artefactos producidos en el desarrollo de software OO.

los elementos de esta etapa (objetivos e hipótesis) se basarán todas las etapas siguientes. Como resultado de esta etapa se generan los requisitos que debe satisfacer la métrica.

- *Creación*: Esta etapa está dividida en tres sub-etapas:

- *Definición de métricas*: El primer paso es la propuesta de métricas. La definición es realizada teniendo en cuenta las características del sistema que queremos medir y la experiencia de diseñadores de estos sistemas. En esta etapa se aconseja utilizar una aproximación como GQM (Goal-Question-Metric) [1] para la especificación de los objetivos de las métricas.

- *Validación Teórica*: Esta validación nos ayuda a asegurar que las métricas realmente miden los atributos que se pretenden medir. Además nos permite conocer la escala a la que pertenece una métrica y así determinar que operaciones estadísticas se pueden realizar con las valores de las métricas. Hay dos tendencias principales para realizar validación teórica de las métricas: una basada propiedades [3], [18] y otra basada en la teoría de la medida [15], [19], [20].

- *Validación Empírica*: La meta de esta etapa es probar la utilidad práctica de las métricas propuestas. Existen varias estrategias para realizar estudios empíricos: experimentos, casos de estudio y encuestas.

  Esta etapa es una de las más importantes y larga, que abarca un proceso iterativo en el cual las métricas pueden ser redefinidas o descartadas dependiendo de su validación teórica y empírica. Como resultado de esta etapa deberíamos obtener una métrica válida tanto teórica como empíricamente.

- *Aceptación*: Una vez obtenida una métrica válida, suele ser necesario pasar por una etapa de aceptación de la métrica en la que se harán pruebas en entornos reales, de manera que podamos comprobar si la métrica cumple los objetivos deseados dentro del campo de aplicación real.

- *Aplicación*: Una vez que tengamos una métrica aceptada, la utilizaremos dentro del entorno para la que fue diseñada.

- *Acreditación*: Es la última etapa del método, que discurre en paralelo con la fase de aplicación y tiene como objetivo el mantenimiento de la métrica, para que pueda ser adaptada al entorno cambiante de aplicación. Como consecuencia de esta etapa, puede ocurrir que una métrica sea retirada, porque ya no resulte útil en el entorno en el que se aplica o que sea reutilizada para reiniciar el método de nuevo.

3. DEFINICIÓN DE MÉTRICAS PARA LA COMPLEJIDAD ESTRUCTURAL DE EXPRESIONES OCL

Debido a la limitación de espacio en esta sección sólo resumiremos el estado de nuestra investigación. En primer lugar hemos definido un conjunto de métricas para la complejidad estructural de expresiones OCL que requieren de los modeladores técnicas cognitivas de "tracing" para su comprensibilidad (ver tabla 1).
Y luego hemos validado teóricamente las métricas propuestas siguiendo el marco formal basado en propiedades propuesto por Briand et al. [3], llegando a la conclusión que las métricas que proponemos son métricas de acoplamiento con excepción de las métricas: DN que es de longitud, WN y NOC que son métricas de tamaño. Además NME cumple ser una métrica de acoplamiento y tamaño.

4. CONCLUSIONES Y TRABAJO FUTURO

En el presente trabajo se ha presentado el

Siglas	Nombre	Definición
NNR	NUMERO DE RELACIONES NAVEGADAS.	El número de relaciones navegadas en una expresión. Una relación es navegada cuando se utiliza en una expresión el nombre de rol del extremo opuesto de una relación que vincula la clase donde se define la expresión con otra clase del diagrama (o eventualmente si el rol es omitido en el modelo, el nombre de la clase asociada al extremo opuesto). Si una relación es navegada dos veces, la contaremos una sola vez.
NAN	NUMERO DE ATRIBUTOS REFERIDOS POR MEDIO DE NAVEGACIONES.	El número de atributos referidos a partir de las navegaciones. Una navegación puede ser utilizada para referir a un atributo de otra clase o interfaz. Los atributos referenciados más de una vez son contados una sola vez.
WNO	NUMERO PONDERADO DE OPERACIONES REFERIDAS POR MEDIO DE NAVEGACIONES.	El número ponderando de operaciones que son referidas a partir de navegaciones. Las operaciones son ponderadas por el número de parámetros actuales de la operación (sólo es necesario especificar los parámetros de entrada y entrada/salida al invocar una operación [16]) y por la cantidad de valores resultantes accedidos (esto es, el resultado de la operación más la cantidad de parámetros de salida ó entrada/salida de la operación).
NNC	NÚMERO DE CLASES Ó CLASES DE ASOCIACIÓN Ó INTERFACES A LAS CUALES SE HA NAVEGADO.	El número de clases o interfaces a las cuales es posible navegar. Si la clase que contiene la expresión contiene una relación reflexiva, y esta relación es navegada, se contará una sola vez a la clase. Además una clase puede ser alcanzable desde una clase inicial a partir de diferentes formas de navegación (es decir siguiendo diferentes relaciones), por ello si una clase es utilizada en más de una navegación esta clase será contada una sola vez.
WNM	NUMERO PONDERADO DE MENSAJES.	El número de mensajes definidos en una expresión [16], donde los mensajes están ponderados en función de sus parámetros.
NPT	NUMERO DE PARÁMETROS CUYOS TIPOS SON CLASES O INTERFACES	Esta métrica es utilizada especialmente en pre- y post-condiciones para un operación, y toma en cuenta los parámetros formales de la operación que son utilizados en la definición de la expresión, cuyos tipos representan clases o interfaces definidas en el diagrama de clases.
NUCA	NÚMERO DE ATRIBUTOS DE UNA CLASE DE UTILIDAD.	El número de atributos referenciados a partir de una clase de utilidad. Una clase de utilidad es un tipo de valor (del inglés 'value type') definido como un nuevo tipo en un modelo UML y utilizado como si fuera un tipo básico [17].
NUCO	NÚMERO DE OPERACIONES DE UNA CLASE DE UTILIDAD.	El número de operaciones referenciadas a partir de una clase utilidad.
WNN	CANTIDAD PONDERADA DE NAVEGACIONES.	Debido a que las navegaciones pueden involucrar la definición de nuevas navegaciones, se pondera con esta métrica a las navegaciones que son definidas en forma recursiva.
DN	PROFUNDIDAD DE LA NAVEGACIONES.	La profundidad de un árbol de navegación. La explicación de la construcción del árbol escapa al propósito de este artículo.
WCO	CANTIDAD PONDERADA DE OPERACIONES COLECCIÓN.	Número de operaciones colección ponderadas.

Tabla 1: Métricas para la complejidad estructural de una expresión OCL, que requieren emplear técnicas de "tracing".

estado de nuestro trabajo de investigación con respecto a la medición de la complejidad estructural de expresiones OCL. Se han aplicado las etapas de Identificación y Creación descritas anteriormente, con excepción de la validación empírica, la cual estamos actualmente planificando con el objetivo de evaluar si las métricas propuestas son realmente útiles en la práctica. Además están pendientes las etapas de Aceptación, Aplicación y Acreditación.

Los actuales avances han sido presentados en un taller internacional que está en proceso de revisión. También prevemos la publicación de los resultados obtenidos y los futuros, en congresos nacionales e internacionales.

AGRADECIMIENTOS

Esta investigación es parte del proyecto DOLMEN financiado por la Subdirección General de Proyectos de Investigación - Ministerio de Ciencia y Tecnología (TIC 2000-1673-C06-06), por el proyecto VII-J-RITOS2 (Red Iberoamericana de Tecnologías de Software para la Década del 2000), y el proyecto

UNComa 04/E048 (Modelado de Componentes Distribuídos Orientados a Objetos).

REFERENCIAS

[1] Basili V., Rombach H., The TAME project: towards improvement-oriented software environments. In IEEE Transactions on Software Engineering 14(6) 758-773, 1998.

[2] Briand L. C., Morasca S., Basili V.. Defining and validating measures for object-based high level design. IEEE Transactions on Software Engineering. Vol. 25 Nº 5, 722-743.

[3] Briand L.C., Morasca S., Basili V., "Property-based software engineering measurement", IEEE Transactions on Software Engineering, 1996, 22, (1) pp. 68-85.

[4] Briand. L. C., Wüst J., Lounis H. A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study. 21st Int´l Conf. Software Engineering, Los Angeles, 345-354. 1999.

[5] Calero C., Piattini M. y Genero M., "Method for obtaining correct metrics", Proc. of the 3rd International Conference on Enterprise and Information Systems (ICEIS`2001), 2001, pp. 779-784.

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

[7] Cantone G., Donzelli P., "Production and maintenance of software measurement models", Journal of Software Engineering and Knowledge Engineering, 5, 2000, pp. 605-626.

[8] Cartwright M. An Empirical view of inheritance. Information and Software Technology, Vol. 40 Nº 14, 795-799. 1998.

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

[10] Fenton N. and Pfleeger S., Software Metrics: A Rigorous and Practical Approach. Chapman & Hall, London, 2nd Edition. International Thomson Publishing Inc. 1997.

[11] Genero M., Piattini M., and Calero C., "Early Measures For UML class diagrams", L´Objet, 6(4), Hermes Science Publications, 2000, pp. 489-515.

[12] Gogolla M. and Richters M. Expressing UML Class Diagrams Properties with OCL. In Tony Clark and Jos Warmer, editors, Advances in Object Modelling with the OCL, pages 86-115. Springer, Berlin, LNCS 2263, 2001.

[13] Harrison R., Counsell S., Nithi R.. Coupling Metrics for Object-Oriented Design. Metrics 1998, Metrics 1998, 150-156. 1998.

[14] Object Modeling with the OCL. The Rationale behind the Object Constraint Language. Lecture Notes in Computer Science 2263. Springer.

[15] Poels G. and Dedene G., "Distance-based software measurement: necessary and sufficient properties for software measures", Information and Software Technology, 2000, 42, (1), pp. 35-46

[16] Response to the UML 2.0 OCL RfP (ad/2000-09-03) . OMG Document ad/2002-05-09. Revised Submission, Version 1.5, June 3, 2002.

[17] Warmer J. and Kleppe A.. The Object Constraint Language. Precise Modeling with UML. Object Technology Series. Addison-Wesley.

[18] Weyuker E.J., "Evaluating software complexity measures", IEEE Transactions on Software Engineering, 1988, 14(9) pp. 1357-1365

[19] Whitmire S.A., Object Oriented Design Measurement Ed. Wiley, 1997

[20] Zuse H., A Framework of Software Measurement Walter de Gruyter, 1998