Metricas aplicables al Diseño Orientado a Objetos

Rosana Montes Soldado. Granada Febrero 2000

Resumen

En el ámbito del desarrollo de software cada vez más se hace necesario el uso de medidas de calidad de software debido al incremento en la demanda de metricas centradas en los procesos desarrollados. Las métricas desarrolladas en anteriores investigaciones, a pesar de haber contribuido al campo de la comprensión del desarrollo software de procesos, han sido el centro de duras criticas, incluyendo la falta de fundamentos teóricos. Posteriores investigaciones a las de Wand and Weber dieron lugar a la elección de una base teorética para las métricas denominada ontología de Bunge. Frente al conjunto de principios de medida de Weyuker, Chidamber y Kemerer desarrollan seis medidas que evaluan analíticamente. A su vez comentaremos las herramientas que han automatizado la recolección de datos y el muestreado empírico de dichas métricas, dando a los desarrolladores una forma de utilizar las métricas para mejorar los procesos de desarrollo de software.

1 El problema de investigación

Se puede decir que hay dos tipos de críticas que han sido recibidas las actuales métricas software:

1. Críticas sobre la parte teorética en el sentido de que las medidas software convencionales deben ser aplicadas al software tradicional y no al software orientado a objetos.

2. El enfoque Orientado a Objetos se centra en modelar el mundo real en términos de sus objetos, lo cual está en total contraste con los más antiguos y tradicionales enfoques, que enfatizan la visión orientada a la función, que separa datos y procedimientos.

2 Teoría Base para derivar las Métricas DOO

Hay muchos autores que han dado su visión de las características esenciales del Diseño Orientado a Objetos como son Budd y Booch. Este último presenta unas líneas muy claras de los pasos realizados en el diseño de procesos orientados a objetos.

1. Identificación de las clases, y por tanto de los objetos (instancias de estos)
2. Identificación de la semántica de la clase. Define el comporamiento del objeto.
3. Identificación de las relaciones entre clases (y objetos)
4. Implementación de las clases

Los principios ontologicos propuestos por Bunge en su obra "*Treatise on Basic Philosophy*" forman la base de conceptos sobre objetos. Los objetos son definidos independientemente de consideraciones sobre implementación y se tienen en cuenta aspectos sobre encapsulación, independencia y herencia. De acuerdo con esta ontología, el mundo es visto como composición de cosas, referidas como individuos sustanciales y conceptos. La principal noción sobre estos individuos sustanciales es que poseen propiedades propias de forma inherente. A su vez un observador podrá asignar caracteristicas a un individuo, pero se considerará atributos y no propieades a estas.

Un objeto no es tan solo un conjunto de métodos, sino una representación del dominio de la aplicación que incluye métodos y variables de instancia que el diseñador asigna a ese objeto.

Bunge describe todos estos conceptos en términos de conjuntos y da definiciones para las siguientes propiedades fundamentales:

Concepto	Comentario
Acoplamiento	El acoplamiento entre clases (debe ser mínimo)
Cohesión	Grado de similaridad en una clase (debe ser máximo)
Complejidad	Complejidad propiedades (operaciones y atributos)
Ámbito propiedades	Extiension de la influencia de una propiedad
Comunicacion	Conjunto de mensajes a los que el objeto puede responder
Composición	Combinación de clases

Otros conceptos se expresan en términos del grafo de herencia, como son la Profundidad de la Herencia y el Número de hijos.

2.1 Criterios de evaluación de Medidas

Weyuker dio una lista formal con nueve propiedades deseables en las métricas de software. Todas las propieades exigen ciertos requerimientos. Se dan condiciones necesarias, pero no suficientes:

1. *Noncoarseness*: dada una clase siempre se podrá encontrar otra con distinta medida de complejidad.
2. *Nonuniqueness:* dos clases pueden ser equivalentes en complejidad. Están al mismo nivel de granularidad.
3. *Design Details are Inportant:* aunque dos clases realicen la misma función, pueden hacerlo con complejidades diferentes.
4. *Monotonicity:* la propiedad de monotonía indica que la combinación de métricas no pueder dar un resultado inferior que la suma de las métricas de forma individual.
5. *Nonequivalence of Interaction:* las interacciones no tienen por qué ser equivalentes
6. *Interaction Increases Complexity:* cuando dos clases se combinan, la interacción entre ellas incrementa la complejidad

El resto de propiedades no se aplican.

Como resultado S.R. Chidamber y C.F. Kemerer propusieron un conjunto de métricas conocidas como métricas C-K. Gracias a las medidas podemos pasar de un sistema empírico D a un sistema formal F. Conoceremos por del nombre de *viewpoint* a una relación empírica dada en D.

3 Medidas resultantes

Metrica1 ***Weighted Methods Per Class*** *(WMC)*

Esta medida está relacionada directamente con la definición dada por Bunge de la complejidad en términos de la cardinalidad de un conjunto de propiedades. En este caso se tiene en cuenta la complejidad de los métodos y esta medida se calcula como la sumatoria de estos.

$$WMC = \sum \mathrm{c}i$$

- El numero de métodos y la complejidad de estos es un predictor de cuánto tiempo y esfuerzo es requerido para desarrollar y mantener la clase

- A mayor número de métodos, mayor es el potencial de impacto sobre sus hijos.
- Clases con un número elevado de métodos tienden a ser de propósito específico, limitando la posibilidad de reutilización.

Metrica 2 *Depth of Inheritance Tree (DIT)*

Según la noción de Bunge de ámbito de las propiedades, se puede observar el grafo de herencia y dar una medida, que extendiendonos a casos en los que entrara la herencia múltiple, se define como la máxima longitud que hay desde un nodo cualquiera al nodo raíz del arbol.

De esta medida se obtienen varias conclusiones:

- Cuanto más profundo encontremos a una clase en la jerarquía, mayor número de métodos habrá heredado y por tanto podemos considerar que será más complejo determinar el comportamiento de la clase.
- Cuanto más profundo encontremos a una clase en la jerarquía, mayor será el potencial de reutilización de dicha clase y de los métodos heredados.
- Un árbol profundo implica la existencia de un número elevado de clases y métodos, que harán más complejo el proceso de diseño.

El valor DIT obtenido puede determinar el tipo de diseño realizado. Dos posibilidades: *'top heavy'* clases concentradas junto al nodo raíz, y *'bottom heavy'* clases mayoritariamente como nodos hoja del árbol.

Metrica3 *Number of Children (NOC)*

El número de hijos de una clase es una medida relacionada también con la noción de ámbito de las propiedades, que indica cuántas subclases heredarán de la clase ancestro. Con esta medida, se obtienen las siguientes conclusiones:

- A mayor número de hijos, mayor será la reutilización de código ya que se está haciendo principalmente un uso de la herencia.
- Una clase con un alto número de hijos, se considerá mucho más en el diseño debido a su influencia y requiere un mayor numero de test en la fase de pruebas.
- Un número pequeño de hijos puede indicar en ciertos casos, una falta de comunicación entre los desarrolladores que desaprovechan oportunidades de reutilización y sugiere también que no se a adoptado completamente el uso de la herencia en el diseño.

Metrica 4 ***Coupling between object classes*** *(CBO)*

Relaciona la noción de que un objeto está acoplado a otro si instancias de la primera clase, utilizan métodos o variables de instancia de la los objetos de la otra clase. Esta situación no es nada deseable en el paradigma orientado a objetos, entre otras causas por las siguientes:

- Cuanto más independiente es una clase, es más facil para otra aplicación de reutilizar dicha clase.
- Si hay muchas dependencias entre las clases que conforman nuestro sistema, es facil ver que cambios en una parte tendrán amplias consecuencias. Esta sensibilidad del sistema ante cambios hara mucho más dificultoso las tareas de mantenimiento realizadas.
- Cuanto más alto sea el acoplamiento más riguroso ha de ser las pruebas requeridas sobre las clases involucradas.

En ocasiones esta métrica se ve afectada por el lenguaje orientado a objetos utilizado. En lenguajes como Smalltalk en donde 'todo' es un objeto, cualquier interacción entre entidades en tiempo de ejecución se ve realizada como un paso de mensajes, e intervendrá en la métrica CBO. C++ sin embargo, puede realizar control de flujo (for, while ...) y operaciones entre escalares sin involucrar objetos, y por tanto dará un valor para CBO inferior.

Esta medida está relacionada con la anterior. Suele coincidir un valor bajo de NOC con un valor bajo para CBO en respuesta a un limitado uso de la herencia. Un valor alto de NOC por ende de CBO indica al diseñador que posiblemente se esté perdiendo la integridad de la jerarquía y nos encontramos con interconexione innecesarias. En tal caso, sería mejor re-evaluar la porción de diseño realizada.

Metrica 5 ***Response For a Class*** *(RFC)*

La respuesta de una clase es el conjunto de métodos que potencialmente pueden ser ejecutados por el objeto en respuesta a un paso de mensaje. Considera el conjunto de métodos de la clase y todos aquellos métodos que pueden ser llamados desde un método cualquiera:

$$\text{RFC} = |RS| = \{M\} \bigcup \{Ri\}$$

Podemos considerar también esta medida como el potencial de cominicación entre una clase y cualquier otra clase. A su vez sabemos que:

- Si el valor de esta métrica es alto, nos está indicando que puesto que el número posible de métodos a invocar será alto, necesitaremos montar un sistema de pruebas y depuración que exigirá un nivel de comprensión del sistema elevado.
- A mayor respuesta de una clase, mayor complejidad de esta.
- Un número pequeño de clases, generará usualmente un alto número de métodos.

Metrica 6 ***Lack of Cohesion in Methods*** *(LCOM)*

Esta métrica utiliza la noción de grado de similaridad entre métodos. Utilizando conjuntos de métodos para cada clase, se calcula una función llamada similaridad $\alpha()$ ue me dá las intersecciones entre dos conjuntos de métodos de dos clases cualquiera. LCOM es un valor que cuenta el número de pares de métodos cuya similaridad es cero ($\alpha()$ conjunto vacío), menos el número de pares de métodos cuya similaridad es distinta de cero. Esto nos da una médida de cómo son de dispares los métodos. Se pueden obtener las siguientes conclusiones.

- La cohesión entre métodos de una clase es deseable en el paradigma orientado a objetos, puesto que promociona el empleo de encapsulación.
- La falta de cohesión en una clase indica que probablemente la clase necesite ser dividida en dos o más clases
- Una cohesión baja incrementa la complejidad y son más complicadas de probar.
- Un valor de LCOM alto indica que los métodos de la clase son dispares en cuanto a funcionalidad.
- Las clases con valor LCOM alto son menos predecibles que las que dan un valor bajo.

3.1 Comentarios

Estas seis métricas han sido diseñadas para medir los tres primeros pasos dados por Booch en su definición de Diseño Orientado a Objetos:

Medida	Identificacion	Semantica	Relaciones
WCM	X	X	
DIT	X		
NOC	X		
RFC		X	X
CBO			X
LCOM		x	

Todas las métricas satisfacen la mayoría de las propiedades descritas por Weyuker con tan solo una excepción importante. La propiedad 6 (las interacciones incrementan la complejidad) no se ve reflejada por ninguna de las medidas. La intención de esta propiedad es permitir la posibilidad del incremento de la complejidad debido a interacciones. Esto implica que una medida de esta complejidad aumentaría su valor si la clase se subdivide, en lugar de reducirse. Esta idea en el ámbito del diseño orientado no parece funcionar, por lo que no satisfacer esta propiedad se puede ver como beneficioso.

A su vez la medida DIT de profundidad del grafo de herencia no cumple la propiedad 4 (monotonía) tan solo en el caso de que las dos clases tengan una relación *ancestro-descendiente*, dado que la distancia del raíz al ancestro no puede ser mayor que la de uno de sus descendientes.

Salvo lo indicado, este conjunto de métricas pueden ser utilizadas por los diseñadores y los gestores de proyecto como revisión del material de diseño de la aplicación y como control sobre la evolución de esta.

Estas seis métricas han sido aplicadas en algunos proyectos de la vida real, algunos de cuyos ámbitos son:

- Sugerencias a los gestores de proyecto, en orden de detectrar las oportunidades de reutilización
- Recuperación de posibles voilaciones de la filosofía de diseño
- Facilita la revisión del material del diseño y ejercita una medición del control realizado sobre la evolución de la aplicación orientada a objeto.
- Estudio de diferencias entre varios entornos y lenguajes de programación orientados a objeto. La extensión más obvia se centra en la investigación de cómo se ajustan estos indicadores de medida en otras fases del desarrollo como son el diseño, esfuerzo de prueba y mantenimiento, calidad y desempleño del sistema.

4 Resultados posteriores a estas métricas

Una vez presentadas estas métricas por Chidamber y Kemerer[1], parece lógico que la atención de los estudios se centrasen en compobar en qué grado estas medidas se ajustan a su propósito de arrojar luz al proceso de diseño. Sharble y Cohen ofrecieron un análisis de cómo se empleaban las métricas CK por la Boeing Computer Services, evaluando diferentes metodologías Orientadas a Objetos [8]. Li y Henry analizaron dos sistemas comerciales conocidos como UIMS y QUES. Encontraron que las métricas CK explicaban el incremento de la varianza en el esfuerzo de mantenimiento explicado mediante medidas de tamaño tradicionales [3]. Esta búsqueda es significativa, poniendo de relieve que las métricas de complejidad propuestas con anterioridad han sido duramente criticadas debido a su dominio procedural [9].

Chidamber y Kemerer añadieron en su trabajo un estudio empírico de datos procedentes de dos organizaciones comerciales y sugirieron cómo podían ser utilizadas las métricas como ayuda para suavizar los esfuerzos dedicados en el proceso de diseño orientado a objetos [6]. En dicho documento se sugirió cómo debían identificar los gestores de proyecto los valores, para reflejar un óptimo diseño.

Más recientemente, Cartwright y Shepper recogieron datos esta vez procedentes de aplicaciones relacionadas con las telecomunicaciones en el Reino Unido [7]. Utilizando un subconjunto de las métricas CK, encontraron una correlación positiva entre la Profundidad del Arbol de Herencia (DIT) y el número de problemas expresados por los usuarios.

En Dinamarca, Soren Nielsen exploró a su vez las implicaciones que presentan valores elevados para las métricas CK [4]. En su informe incluyó resultados preliminares basados en estudios de modelos de diseño de 56 clases con 170 métodos y 150 atributos. De aquí se identificó dos clases con valores elevados de CBO y LCOM, y que más tarde resultaron ser las de mayor dificultad en la implementación.

Pant, *et al.* coleccionaron datos de todo tipo de clases. El conjunto lo formaban 69 clases de dominio público y 25 clases desarrolladas de forma interna, en su investigación de reutilización de compones orientados a objetos generalizables [5]. Encontraron correlaciones significativas entre las métricas probadas y otras medidas de complejidad como SLOC y la complejidad ciclomática.

Acercándonos a la actualidad Basili *et al.* publicaron los resultados de aplicar métricas CK ajustadas para poder predecir propiedades de cualidad en progra-

mas de estudiantes de C++ [2]. Cinco de las seis métricas fueron demostradas de utilidad como predictores, y se demostró que el ajuste de las métricas CK ofrecían mejores resultados que las métricas de código extendidas.

Merece la pena mencionar la obra de Morris [Morris 1989] que en su tesis hizo algunas observaciones importantes sobre código OO y propuso otras métricas candidatas para la medición de la productividad. Las citaremos sin entrar en más detalles:

- Métodos por Clase
- Dependencias en la jerarquía de herencia
- Media del grado de acoplamiento entre objetos
- Grado de cohesión entre objetos
- Efectividad de la Librería de objetos
- Factor de Efectividad
- Grado de reutilización de los métodos heredados
- Media de la complejidad de los métodos
- Granularidad de la aplicación
- Factor de encapsulación de un método (MHF)
- Factor de encapsulación de un atributo (AHF)
- Factor de herencia de un método (MIF)
- Factor de herencia de un atributo (AIF)
- Factor de polimorfismo (PF)
- Factor de acoplamiento(CF)

5 Conclusiones

Estas métricas están basadas en una medición teórica y a su vez refleja los puntos de vista de aquellos desarrolladores experimentados de software orientado a objetos. Al evaluar estas medidas contra un conjunto estándard de criterios, se concluye que a) posee un numero considerable de propiedades deseables, y b) sugiere algunos caminos en los que la aproximación orientada a objetos debe

diferir con nuevas y necesarias características de diseño, con respecto de las anteriores métricas tentradas en las técnicas tradicionales de diseño.

Gracias a brindarle de un formalismo teórico de medida, la ontología de Bunge, y un criterio de evaluación dado por Weyuker, uniéndolo a un conjunto de datos empíricos procedentes de desarrolladores profesinales trabajando en proyectos comerciales, el documento presentado en Junio de 1994 por Shyam R. Chidamber y Chris. F. Kemmer muestra, el nivel de rigor requerido en el desarrollo de métricas utilizables en el diseño de sistemas software.

6 Comentarios Personales

El documento de Chidamber y Kemmer es una buena referencia sobre la descripción de métricas para el DOO. Estas métricas tienen un fuerte soporte teórico y ha quedado demostrado como válidas en proyectos de la vida real. De hecho, estas medidas estan lo suficientemente preparadas como para ser aplicadas al trabajo del dia a dia.

Las métricas para los lenguajes orientados a objetos tienen distintos requerimientos que las utilizadas en lenguajes convencionales y esto es debido a la deferencia de aproximaciones de diseños y a su vez de los mismos lenguajes empleados. Es bueno por tanto, disponer de métricas específicas orientadas a objetos para problemas orientados a objetos específicos.

Por supuesto que el uso de estas seis medidas nos van a aportar veneficios:

- Es específico OO, es decir, tiene en cuenta el ciclo de vida del desarrollo OO
- Llega a un punto de detalle de trabajo que posibilita su aplicación en la vida real
- Permite ser automatizado. Se pueden encontrar herramientas como QMOOD++ que proporciona un conjunto de herramientas de medida de cualidad orientados a entornos C++
- Es capaz de mostrar de donde se han derivado las métricas
- Provado su uso en la práctica

Queda abierto la superación de estas medidas en ciertos aspectos:

- Se requiere entrenamiento y compromiso
- No ha sido comentado cuál es el *benchmark* obtenible para estas medidas

- La recolección de datos es una tarea captadora de tiempo a no ser que se disponga de automatizaciones
- No se tiene en cuenta las diferentes necesidades existentes en distintos tipos de programas.

Considero que tal vez no hay ningún estimador que ayude a evaluar el impacto producido por las diferentes decisiones del diseño orientado a objetos. Una de estas decisiones podría estar entre el uso de herencia múltiple o herencia simple. Claro que esta medida tendría que llegar analizar las relaciones posibles.

Otra línea de investigación podría encaminarse hacia el análisis del trabajo para diferentes categorías. Esto exigiría recolectar mucha información, y lo más completa posible. A ser posible la información se extraería de otras fases del proceso de desarrollo, como son la especificación, diseño, implementación, ...

La idea principal que quiero resaltar es que lo importante es destacar que el conjunto de estas seis medidas constituye un conjunto válido de métircas formales en el DOO. Estas a su vez han sentado las bases en este campo y ha abierto nuevas posibilidades y futuras investigaciones en este área.

7 Referencias

[1] *Chidamber, S.R and Kemmerer, C.F., Towards a Metrics Suite for Object Oriented Design,* Proc. of the 6th ACM Conference on Object Oriented Programming, Systems, Languages and Appliactions (OOPSLA), 1991, Phoenix, AZ, pp. 197-211.

[2] *Basili, V.R. et al., 'A Validation of Object-Oriented Design Metrics as Quality Indicators'*, IEEE Transactions on Software Engineering, vol. 22, pp. 751-761, 1996

[3] *Li, W. and Henry, S., 'Object-Oriented Metrics that Predict Maintainability'*, Journal of Systems and Software, vol. 5, pp. 49-58, 1985

[4] *Nielsen, S.,* Personal comunication, June 18, 1996.

[5] *Pant, Y. et al., 'Generalization of object-oriented components for reuse: Measurement of effort and size change'*, Journal of Object Oriented Programming, vol. 9, pp. 19-41, 1996.

[6] *Chidamber, S.R. and Kemerer, C.F., 'A Metrics Suite for Object Oriented Design'.* IEEE Transactions on Software Engineering, vol. 20, pp. 476-493, 1994.

[7] *Cartwright, M. and Shepperd, M., 'An Empirical Investigation of Object Oriented Software in Industry'*, Dept. of Computing, Talbot Campus, Bournemouth University Technical Report TR 96/01, 1996.

[8] *Sharble, R.C. and Cohen, S.S., 'The Object-Oriented Brewery: A Comparision of Two Object-Oriented Development Methods'*, Software Engineering Notes, vo. 18, pp. 60-73, 1993.

[9] *Shepperd, M., 'A critique of cyclomatic complexity as a software metric'*, Software Engineering Journal, vol. 3, pp. 30-36, 1988.