Validación de Métricas para Esquemas Conceptuales como Indicadores de Calidad en Modelos Orientados al Objeto

José Romero, Oscar Pastor, J.J. Fons
Universidad Politécnica de Valencia,
Camino de vera s/n, Valencia, España
Correo electrónico: {jromero || opastor ||jjfons}@dsic.upv.es

Resumen

Tradicionalmente, el desarrollo de productos software de calidad ha estado basado en el estudio del código implementado. Puesta de manifiesto la importancia de las etapas tempranas de análisis para el aseguramiento de la calidad, las métricas deben aumentar su nivel de abstracción para referirse a esquemas (modelos) conceptuales. Estas nuevas medidas deben ser validadas teórica y empíricamente. Este planteamiento encaja perfectamente con nuestro enfoque metodológico para la generación automática de aplicaciones a partir de modelos conceptuales orientados al objeto; combinando aspectos formales con notaciones estándar en la industria.

La originalidad del trabajo es establecer y validar relaciones entre las medidas conceptuales y los atributos de calidad del software mediante hipótesis de aseguramiento de calidad. Determinar la calidad de las aplicaciones software generadas automáticamente es analizar las medidas relevantes, capturadas también automáticamente, de un modelo conceptual orientado al objeto. Para ello, se puede utilizar la herramienta CASE que soporta nuestro método.

Palabras Clave
Calidad del software, métricas orientadas al objeto, modelado conceptual, métodos orientados al objeto, generación automática de software.

1. Introducción

Obtener productos software de calidad ha sido el objetivo final perseguido por el método llamado OO-Method[1] desarrollado en nuestro grupo de investigación. Basados en la combinación de métodos formales[2] y notaciones estándar[3] se planteó la creación tanto del método como de una herramienta CASE[4] que le diese soporte. OO-Method se fundamenta en el paradigma objetual y en el paradigma de la programación automática[5]. De esta forma se ha logrado un método de desarrollo muy potente que genera código (aplicaciones completas y no meras plantillas) de una manera automática a partir de modelos conceptuales. Además, estos modelos conceptuales tienen su reflejo en un lenguaje de especificación formal usado como repositorio de datos de alto nivel transparente al usuario. Sin embargo, el usuario percibe el estar trabajando con notaciones ampliamente usadas en la industria.

En los últimos años, la preocupación por el aseguramiento de la calidad nos ha hecho reflexionar sobre cómo dotar de calidad al método en sí y a los productos software que él genera automáticamente. Sobre el primer punto hemos realizado propuestas de definición de un marco de tratamiento de la calidad para OO-Method[6]. En el presente trabajo, se aborda el punto complementario de cómo asegurar la calidad de los productos generados.

En la mayoría de ocasiones, se mide la calidad de un producto basándose en métricas sobre el código que constituye su implementación. Ya que la generación de código en OO-Method es automática, y dada la gran importancia que tienen las primeras etapas de modelado de sistemas en el aseguramiento de la calidad del producto final, se propone subir el nivel de abstracción para definir métricas sobre modelos conceptuales orientados al objeto.

Existen ya propuestas[7] sobre la notación UML que se van acercando a un objetivo similar. Se pretende adaptar estas propuestas, enriquecerlas con las características específicas que contiene OO-Method y su lenguaje de especificación formal OASIS[2], y por último, validarlas teórica y empíricamente.

Partiremos de la experiencia obtenida del trabajo con casos reales resueltos con OO-Method para el establecimiento de hipótesis (de calidad) que permitirán validar empíricamente las medidas propuestas. En un siguiente paso, validaremos de

forma teórica dichas medidas basándonos en marcos formales ya establecidos[8].

Obtenemos así que para evaluar la calidad de un sistema de información, se deben recoger una serie de medidas sobre su correspondiente modelo conceptual. Como la generación de código es automática y el método tiene un diccionario de datos de alto nivel, se puede comprobar fácilmente la calidad de un sistema software. Cabe destacar que una medida es un posible indicador de un problema cuando ésta supera ciertos umbrales. Estos se irán afinando con la realización de un mayor número de casos reales. De igual manera, se propondrá en un trabajo futuro la generalización a otros métodos -bajo ciertas condiciones- de los resultados obtenidos.

En cuanto a la estructuración del presente trabajo, se establecen las hipótesis llamadas de calidad para determinar el conjunto de métricas conceptuales. Después, se plantea cómo deben ser validadas empíricamente basándose en el diseño de experimentos y su análisis estadístico. Luego, se plantea cómo deben ser validadas teóricamente. Para concluir, se presentan las conclusiones y los posibles trabajos futuros.

2. Medidas e hipótesis de calidad

Como se ha avanzado anteriormente, métricas para el aseguramiento de la calidad existen para lenguajes de programación orientados al objeto. Actualmente existen también propuestas a más alto nivel para esquemas conceptuales. Por ello, vamos a escoger una propuesta válida desde el punto de vista formal como base para adaptarla y aplicarla a las particularidades de OO-Method. La definición de las métricas que se presenta en este trabajo cuenta con la base de las observaciones realizadas en los proyectos llevados a cabo por una empresa de software que utiliza OO-Method para la resolución de casos reales. De esta experiencia se ha seleccionado el conjunto de medidas cuya validación empírica planteamos y que serán refinadas conforme se desarrollen más proyectos en la empresa. A partir de estas medidas, seleccionamos aquellas que creemos relevantes para los atributos de calidad de un producto software[9]. Se presentan en forma de hipótesis (llamadas de calidad) la relación entre la medida y el atributo de calidad. El paso siguiente es una vez aceptada la validez empírica, se valida la medida formalmente desde un punto de vista teórico, utilizando un marco matemático bien definido.

Es interesante destacar que aquí se hace especial hincapié en las medidas de un determinado apartado del modelo conceptual OO-Method como es la visión estructural que aporta el modelo de objetos; por ser el campo de investigación en donde se ha trabajado más. Sin embargo, indicaremos también medidas de otros apartados de un modelo conceptual OO-Method. Además, una vez que se defina el proceso de desarrollo de software con OO-Method, se podrán incorporar de forma similar medidas sobre el proceso de desarrollo.

Tomando como punto de partida las métricas existentes tanto para lenguajes como para modelos orientados al objeto, se presentan a continuación un extracto de medidas particularizadas para OO-Method con el único objetivo de dar una idea al lector de la diferenciación con respecto a las métricas existentes en la literatura. Para mayor detalle de los conceptos propios de OO-Method, recomendamos al lector interesado la consulta de las referencias bibliográficas al final de este artículo.

- ***Métricas de Esquema Conceptual:***

Número de *Interacciones Globales*, número de *Agrupamientos (Clústers)*, número de *Vistas* definidas, número de *Eventos Compartidos*, Ratio de uso de herencia, número de *Diagramas de Transición de Estados básicos*, número total de *Disparos*, número de errores de validación, número de avisos de validación

- ***Modelo de Objetos:***

Número de *atributos (constantes, variables, derivados)*, número de *servicios* que componen las *transacciones*, número de *clases servidoras* (que ofertan servicios al resto de clases) que puede activar la clase estudiada como *actora* (*clase activa* cuyas instancias podrán lanzar servicios), número de *interfaces* distintas de su interfaz por defecto, número de *restricciones estáticas*, número de *restricciones dinámicas*

- **Modelo Dinámico**

Número de *estados*, número de *transiciones*, número máximo de *transiciones* para un *estado*, número máximo de *disparos* por clase, número de *disparos* a otro objeto

- **Modelo Funcional**

Número de *evaluaciones (Cardinales, De Estado, De Situación)*, número de *evaluaciones* propias, número de *evaluaciones* heredadas, número de *evaluaciones* redefinidas

Para establecer la relación entre métricas conceptuales y atributos o principios de calidad [9, 10] partiremos de las siguientes hipótesis:

- Con respecto a las guías de modelado
  - Hipótesis 1 (adecuación de construcciones): Cuando un ratio de uso de un elemento es aproximadamente cero, puede existir un problema de inadecuación de construcciones de OO-Method. Es decir, el que un elemento no se use extensivamente puede ser indicador de que el elemento no tiene su semántica bien definida en el método.
  - Hipótesis 2 (adecuación del lenguaje): A mayor ratio de errores de validación / número de clases, puede ser un indicativo de falta de flexibilidad del método a la hora de modelar. Esto lleva a que a mayor flexibilidad en el modelado, menor adecuación del lenguaje de modelado, es decir, se incrementan el número de inconsistencias en los modelos.
  - Hipótesis 3 (diseño sistemático): Un valor absoluto del número de errores de validación mayor que cero denota una falta en el diseño sistemático del sistema.
  - Hipótesis 4 (comparabilidad): El principio de comparabilidad puede deducirse del atributo de calidad llamado portabilidad en las ISO9126 que se explica más adelante.
  - Hipótesis 5 (claridad): Un ratio elevado de relaciones por clases afecta negativamente a la claridad de un modelo. Pueden utilizarse una suma de elementos/ número de clases o bien un valor absoluto del número de clases para determinar la complejidad de un modelo. Un modelo más complejo será menos claro.
  - Hipótesis 6 (eficiencia económica): El principio de eficiencia económica es comparable a la definición del atributo de calidad llamado simplemente "eficiencia" en las normas ISO9126
- Con respecto a los atributos de calidad de ISO9126
  - Hipótesis 7 (funcionalidad): El número de inconsistencias en el análisis realizado con los casos de uso establecidos y los escenarios definidos determinan la funcionalidad del sistema. A mayor número de inconsistencias de este estilo menor funcionalidad en el sistema.
  - Hipótesis 8 (fiabilidad): El número de operadores relacionales en una fórmula y el tamaño máximo del tamaño de una transacción son elementos que inciden directamente sobre la fiabilidad de un modelo analizado con OO-Method.
  - Hipótesis 9 (ergonomía): El número de clases visibles relacionadas puede ser un indicador de la ergonomía a la hora de manipular el modelo. A mayor visibilidad mayor flexibilidad para definir nuevas fórmulas en una clase.
  - Hipótesis 10 (eficiencia): El número de clases introducidas o borradas en una iteración puede ayudar a determinar si un modelo evoluciona eficientemente hacia los requisitos propuestos. A menor variación de clases por iteración se está más cerca de una solución a medida del cliente. Una variación grande, puede indicar que ha habido un error a la hora de tomar requisitos.
  - Hipótesis 11 (mantenimiento): La facilidad de mantenimiento de una aplicación vendrá en función de las medidas de complejidad. A mayor complejidad (por ejemplo número de clases visibles) mayor esfuerzo de mantenimiento habrá que hacer en el modelo cuando se requiera hacer un cambio.
  - Hipótesis 12 (portabilidad): La portabilidad de un modelo vendrá determinada por el número de elementos no representados con la notación estándar UML.

Una vez detalladas las hipótesis de calidad es necesario validarlas empíricamente, y por supuesto, formalmente. Estos dos aspectos son los que tratamos en los siguientes apartados. Sólo en el caso de que se acepten las medidas como válidas tanto empírica como teóricamente, se aceptará la medida como indicador de calidad. Resaltar también que el conjunto de métricas no sólo contiene medidas de la complejidad del modelo, sino que también se aglutinan medidas de concordancia sintáctica y semántica con el propio método y con los requisitos analizados con el método.

3. Modelo de validación empírica

En este punto se propone el modelo basado en el diseño de experimentos que permite validar empíricamente las hipótesis propuestas. Determinando las métricas involucradas en las hipótesis establecemos las medidas significativas para medir la calidad de un esquema conceptual. Cabe recordar que mientras otros trabajos se centran en la probabilidad de encontrar clases propensas a tener errores[11], nosotros asumimos que la generación de código está libre de errores por estar basada en patrones bien definidos y estudiados. Los errores pueden venir al construir el modelo conceptual, pero para este inconveniente existe un proceso de validación sintáctica y semántica implementado en la herramienta OO-Method/CASE que da soporte al método.

Describiendo propiamente el modelo de validación empírica, se debe tener en cuenta los siguientes pasos para validar las hipótesis planteadas:

i) Establecer un caso práctico real a resolver por distintos modeladores.

ii) Seleccionar los participantes involucrados en el experimento. Se determinará el nivel de experiencia de los participantes mediante cuestionarios o entrevistas con el fin de asignar aleatoriamente a distintos grupos los modeladores con mayor experiencia.

iii) Determinar los productos entregables después del proceso de modelado. Es decir el tipo de documentación a entregar: modelo analizado y métricas obtenidas.

iv) Realizar las pruebas oportunas para comprobar que los entregables se ajustan o no a los requisitos planteados. De este modo, en vez de centrarnos en errores de código, nos centramos en la falta de concordancia con los escenarios establecidos.

Después de este planteamiento se realiza el pertinente análisis, utilizando la estadística descriptiva para interpretar los resultados.

Se toman valores de máximo, mínimo, media, mediana, y desviación estándar para cada medida incluida en las hipótesis. A continuación, se realiza un análisis de correlación entre las mismas para comprobar que las hipótesis (sus medidas) son independientes. Finalmente, se establece la relación entre probabilidad de falta de concordancia del modelo con los requisitos en base a un análisis de regresión univariante. Al así proceder, se validan las métricas (y las hipótesis) para el aseguramiento de la calidad del esquema conceptual modelado.

Por ejemplo, al validar la hipótesis de fiabilidad que nos decía que a mayor número de operadores relacionales en las fórmulas de una clase desciende la fiabilidad del sistema generado, haremos lo siguiente: el usuario evaluará de 0 a 10 la fiabilidad del sistema contrastando los requisitos (casos de uso) establecidos y el número de faltas de concordancia encontrados. Teniendo las dos variables (fiabilidad y número de operadores) en forma continua, se establece la recta de regresión y se realiza un análisis viendo el coeficiente de correlación, o bien, por medio de un análisis de residuos. Además, se puede concretar si la relación entre las variables es lineal o no, realizando y observando el diagrama de dispersión obtenido.

También es posible establecer un modelo multivariante en el que comprobar el efecto simultáneo de varias métricas sobre la falta de calidad. Este análisis multivariante utilizará únicamente las variables halladas significativas en el análisis univariante.

4. Validación teórica de las métricas

Para validar formalmente las métricas seleccionadas se propone utilizar el marco definido por Briand et al.[8]. Este marco define de forma precisa qué propiedades matemáticas caracterizan los conceptos usados en la medición del software. Además este marco es aplicable a cualquier artefacto software, y se basa en los conceptos de tamaño, longitud, complejidad, cohesión y acoplamiento. En el caso particular de OO-Method, se trata el Modelo Conceptual como abstracción del software que será generado automáticamente en un proceso que es transparente al usuario. Es decir, que las medidas del software son estudiadas en su representación conceptual. Esto es posible por la homogeneidad que aporta el paradigma de la

orientación al objeto y por la base de patrones conceptuales OO-Method bien definidos que trasladan las componentes conceptuales en las componentes software de la solución a entregar al cliente.

Repasando los fundamentos teóricos del marco, aparece el concepto de Sistema que en nuestro caso es equiparable al de Esquema Conceptual. Un Sistema(Esquema Conceptual) S se define como un par <E,R> donde E representa el conjunto de elementos de S, y R es una relación binaria sobre E ($R \subseteq ExE$) representando las relaciones entre los elementos del sistema(Esquema Conceptual).

Un módulo (equiparable en OO-Method al concepto de clase) se define dado un sistema S=<E,R> como m=<Em, Rm> si y sólo si $Em \subseteq E$, $Rm \subseteq Em \times Em'$ y $Rm \subseteq R$.

Las relaciones intramodulares(intraclase) se definen como InputR(m)={<e1,e2>$\in$ R| e2$\in$Em and e1$\in$E-Em}.

Las relaciones intermodulares(interclase) se definen como OutputR(m)= {<e1,e2>$\in$ R| e1$\in$Em and e2$\in$E-Em}.

La caracterización de las medidas se hace en base a propiedades bien definidas, por ejemplo, para una medida de tamaño, se comprobarán las propiedades de no negatividad, la existencia de valor nulo, y la adición.

Así pues, para validar la medida OO-Method del número de relaciones de agentes de una clase tendremos que ver que:

i) El número de servicios que es posible activar de una clase servidora por parte de una actora será siempre mayor o igual a cero; es decir, no tiene sentido la negatividad.

ii) Si no hay relaciones de agente entonces el Número de Relaciones de Agente (NRA) será cero.

iii) Si dos clases servidoras {A,B} se fusionan conceptualmente en el modelado en una sola {C} entonces NRA(A)+ NRA(B) = NRA(C).

De forma similar se validarían formalmente el resto de las métricas de tamaño para OO-Method. Para el resto de medidas de longitud, cohesión y acoplamiento se sigue el esquema presentado en el método de Briand que aquí no detallamos por no ser el objetivo principal de este trabajo.

5. Conclusiones

En el camino seguido para utilizar métricas conceptuales que sean validadas formal y empíricamente para el aseguramiento de la calidad del software, se ha establecido un subconjunto de medidas para el método OO-Method partiendo de las medidas existentes en el estado del arte. En un segundo paso, se establece una serie de hipótesis de calidad que sirven de base para relacionar las medidas conceptuales con los atributos de calidad que se les supone a los productos software. Finalmente, se plantea de manera esquemática el modelo para validar cada una de las hipótesis de calidad. Esta validación se realizará de forma empírica, y las medidas implicadas se validarán también teóricamente dentro del marco formal que se ha comentado. Por razones de espacio se detalla el proceso, y no cada una de las correspondientes validaciones de hipótesis y medidas de la calidad.

La utilidad del estudio realizado reside en la facilidad para determinar la calidad de un producto software generado automáticamente consultando el conjunto de medidas validadas (obtenidas también de forma automática utilizando una herramienta CASE que soporte modelos conceptuales OO-Method).

Como trabajo futuro destacamos la generalización de las medidas conceptuales para otros métodos de generación automática de código basados en modelos conceptuales orientados al objeto. Además, estas medidas deben ser sometidas a un mayor número de casos reales, lo que permitirá en un futuro poder refinar los umbrales de dichas medidas.

6. Referencias

[1] Pastor, O.; Romero, J.; Pelechano, V.;Insfran, E.;Merseguer,J. *OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods.* CAISE-97.

[2] Pastor, O.;Hayes,F.;Bear,S. *OASIS:An OO Specification Language.* Proc. of CAiSE-92 Conference, Lncs (593), Springer-Verlag 1992. pags: 348-363.

[3] Booch,G.;Rumbaugh,J.;Jacobson,I. *Unified Modeling Language (UML summary). Version 1.0 January 1997.* Rational Software Corporation.

[4] Romero, J.; Merseguer,J.; Barberá J.; Pastor O. *Una herraminienta de generación automática de Sw.* Jornadas de ingeniería del SW IDEAS'98. Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil

[5] Balzer, R. et al. *Software Technology in the 1990s: Using a New Paradigm.* IEEE Computer, Nov. 1983.

[6] Romero, J.; Pastor O. *Propuesta metodológica para el tratamiento de la calidad en la producción de software a partir de modelos conceptuales.* Jornadas de ingeniería del SW IDEAS'00. Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET). México.

[7] Genero, M.; Piattini, M; Calero, C. *Métricas para jerarquías de agregación en diagramas de clases UML.* Jornadas de ingeniería del SW IDEAS'00. Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET). México.

[8] Brian, Lionel C; Morasca, Sandro; Basili, Victor R. *Property-based software engineering measurement.* IEEE Transactions on Software Engineering, Vol 22, No 1, January 1996.

[9] International Organization for Standarization [online]. December 1998. From World Wide Web: http://www.iso.ch

[10] Schuette.Reinhard; Rotthowe, Thomas; *The Guidelines of Modeling - An Approach to Enhance the Quality in Information Models.* Proceedings of the International Conference on the Entity Relationship Approach (ER). Singapore 1998

[11] Basili, Victor R.; Brian, Lionel C; Melo, Walcélio L. *A validation of Object-Oriented Design Metrics as Quality Indicators.* IEEE Transactions on Software Engineering, Vol 22, No 10, October 1996.