Una Propuesta conforme a MOF para la Modelización de la Calidad del Software¥

Xavier Burgués[1], Xavier Franch[1], Josep M. Ribó[2]

[1] Universitat Politècnica de Catalunya (UPC)
c/ Jordi Girona 1-3 (Campus Nord, C6) E-08034 Barcelona (Catalunya, Spain)
{diafebus, franch}@lsi.upc.es
[2] Universitat de Lleida (UdL)
C. Jaume II, 69 E-25001 Lleida (Catalunya, Spain)
josepma@eps.udl.es

Resumen. En trabajos previos hemos propuesto formalizar los aspectos de calidad de los artefactos software mediante tres tipos de modelos estructurados jerárquicamente. En este artículo hemos integrado nuestra propuesta en el metamodelo de UML y, en concreto, en la arquitectura MOF de 4 niveles. Hemos aplicado una metodología de extensión del metamodelo UML y hemos introducido la noción de asociación inducida por un metamodelo.

1. Introducción

Diversos autores coinciden en la importancia de definir ontologías, modelos conceptuales, modelos de datos o similares, para definir con precisión los conceptos, procesos, lenguajes y herramientas necesarios para razonar sobre la calidad del software [Gar+04, KHL01, Ols+02] definiendo un marco de trabajo que permita analizar las muchas propuestas existentes.

En [BF03] presentamos nuestra propuesta (una jerarquía de modelos de calidad), resumida en la sección 2. Este artículo acomoda la propuesta en la arquitectura MOF de 4 niveles [MOF04]: el modelo genérico se define en M2 y los modelos de referencia y de dominio en M1. Para ello efectuamos una extensión del metamodelo de UML [UML04]. La parte central de esta tarea recae en una metodología de extensión de metamodelos de UML que hemos definido en [RF02] y que aquí resumimos en la sección 3. Esta metodología ha sido enriquecida con una nueva construcción (que llamamos *asociación inducida*) presentada en la sección 4. En las secciones 5 y 6 mostramos el resultado de la aplicación de la metodología en nuestra propuesta.

2. Una jerarquía de modelos de calidad

En [BF03] presentamos un marco de referencia para la definición de modelos de calidad basado en UML, notación suficientemente expresiva, inteligible gracias a su in-

¥ Este trabajo se ha desarrollado en el marco del proyecto CICYT TIC2001-2165.

terfaz gráfica, ampliamente usada y con herramientas asociadas. Dicho marco está estructurado como una jerarquía de tres tipos de modelos conceptuales:

- Modelo genérico. La raíz de la jerarquía. Introduce los conceptos fundamentales que están presentes en toda propuesta referente a los modelos de calidad del software, tales como "modelo de calidad", "artefacto" y "métrica". Su alto nivel de abstracción le permite ser usado en diversas actividades de la ingeniería del software que requieran modelos de calidad.
- Modelos de referencia. Proveen interpretaciones particulares de los conceptos del modelo genérico en contextos particulares, tales como especificación de requisitos, arquitecturas de software, etc. Por ejemplo, este modelo puede incluir conceptos del estándar ISO/IEC 9126 de calidad [ISO01] y nociones de métricas según la teoría recogida en [FP98, Zus98].
- Modelos de dominio. Refinan los conceptos de un modelo de referencia para adaptarlos a un dominio particular. Ejemplos de dominios son: sistemas en tiempo real, servicios web, bibliotecas de componentes, bases de datos, etc.

Los tres tipos de modelos están estructurados en cuatro partes:

- Contexto. Contiene información sobre los dominios a los que se dirigen los modelos de calidad, sobre la estructura de los artefactos a medir, y sobre el entorno en el que operan (un tipo de organización, un proyecto concreto, etc.). Los dominios pueden estructurarse en taxonomías [GV95]. Los artefactos pueden ser agregaciones o composiciones de otros.
- Marco conceptual. Engloba los conceptos y relaciones que forman los modelos de calidad y los requisitos sobre calidad. Los conceptos manejados provienen de estándares de calidad [IEE92, ISO01], ontologías y modelos de datos sobre calidad [Gar+04, KHL01] y catálogos de factores de calidad y requisitos [Fir03, KKP90].
- Métricas. Definimos los tipos de métricas usadas para medir los elementos definidos en el modelo y analizar la satisfacción de los requisitos. La teoría de métricas [FP98, Zus98] y de nuevo estándares y ontologías proporcionan información útil.
- Lenguaje. Parte relacionada con la notación usada para expresar los elementos de las tres partes precedentes. Puede ser un lenguaje de propósito general adaptable al marco que nos ocupa, como XML o DAML [XML04, DAML04] o bien lenguajes específicos de la calidad como QRL o GlueQoS [RCDT02, Woh+04].

3. Metodología de extensión del metamodelo de UML

Definimos nuestra jerarquía de modelos partiendo de una extensión del metamodelo de UML para generar modelos de calidad de los productos de software, con el requisito fundamental de que debe integrarse completamente en la arquitectura de metamodelización de 4 niveles y debe, asimismo, ser coherente con el mecanismo estándar de extensión de UML. Usamos la metodología detallada en [RF02] que consiste en una extensión pesada (*heavyweight*) del metamodelo de UML y su transformación en un *profile UML*, aunque en este artículo nos centramos en presentar la extensión pesada del metamodelo. De todos modos, antes debemos detenernos en un problema de metamodelización que se ha planteado en el diseño de esta extensión y que no estudiamos en el trabajo anteriormente citado [RF02]: el de las asociaciones inducidas.

4. Asociaciones inducidas

En esta sección nos interesa llamar la atención sobre las metaasociaciones (definidas a nivel M2) y las asociaciones (definidas a nivel M1). Para ello consideremos el ejemplo de la figura 1(a), que muestra una parte del metamodelo para formalizar modelos de calidad de productos de software. Se muestran dos metaclases: *ModeloAplicable* (que representa una clase de modelos de calidad como ISO-IEC 9126) y *Cuestión* (que modeliza una forma de expresar condiciones o requisitos; por ejemplo, predicados de primer orden). También se muestra una metaasociación *concierne* entre ellos.

La instanciación de este fragmento tendrá lugar a nivel *M1* tal y como muestra la figura 1(b). En este nivel encontraremos instancias de cualquiera de las dos metaclases presentadas (e.g., *CP1* –predicados de primer orden-, de *Cuestión* y *ISOFNF-AplInformatica* –modelos ISO para aplicaciones informáticas-, de *ModeloAplicable*) y también tendremos una extensión de la metaasociación *concierne* (*e.g.*, el par *<CP1, ISOFNF-AplInformatica>*). Nótese que la metaasociación del metamodelo no implica una asociación entre las instancias. Las *asociaciones inducidas*, definidas a continuación, garantizarán que en toda instancia correcta existirá una asociación a nivel *M1*, *concierne_M1*, entre las clases, tal y como muestra la figura 2.

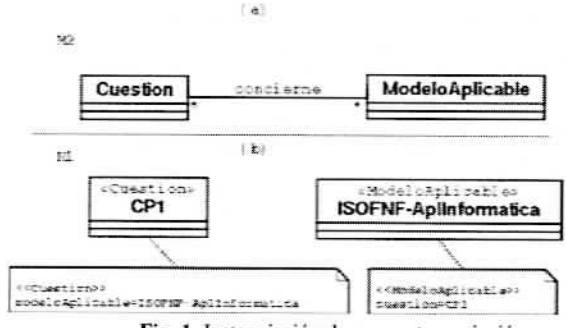

Fig. 1. Instanciación de una metaasociación

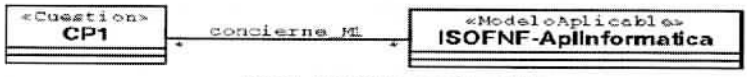

Fig. 2. Asociación *concierne_M1*

La idea consiste en añadir al metamodelo: 1) una metaclase *AssConcierne* que sea subclase de la metaclase *Association* del metamodelo de UML, y 2) una restricción ligada a la metasociación *concierne* que establezca que cualquier modelo a nivel *M1* debe contener una asociación, instancia de *AssConcierne* que conecte cada par de clases que constituyan la extensión de *concierne*. Como consecuencia de esta restricción, se deberá establecer a nivel *M1* la asociación entre las clases *CP1* y *ISOFNF-AplInformatica* que se muestra en la figura 2.

Definición. Asociación inducida por una metaasociación. Consideremos una extensión del metamodelo de UML a nivel *M2* con dos metaclases *MC1* y *MC2* y una

metaasociación entre ellas, *M2A*. *M2A* induce asociaciones a nivel *M1* si la extensión del metamodelo contiene (v. fig. 3):

- Una subclase *M1A* de la metaclase de UML *Association*.
- Una restricción `[c1]` asociada a *M2A* que establezca:

```
[c1]: MC1.allInstances()->forAll(c1|
        MC2.allInstances()->forAll(c2| c2.mc1=c1 implies
          M1A.allInstances()->exists(a|
                a.memberEnd->exists(e|e.class=c1) and
                    a.memberEnd->exists(e|e.class=c2))))
```

- Una restricción asociada a *M1A* que establezca

```
[c2]: M1A.allInstances()->forAll(a|
        a.memberEnd->size()=2 and
        a.memberEnd->exists(e|e.class.oclIsTypeOf(MC1)) and
        a.memberEnd->exists(e|e.class.oclIsTypeOf(MC2)))
```

En el metamodelo podemos denotar de manera más simplificada que una metaasociación induce asociaciones a nivel *M1* con una función booleana *induceAsociación(Metaasociación, Asociación)* con el significado establecido por [c1] y [c2].

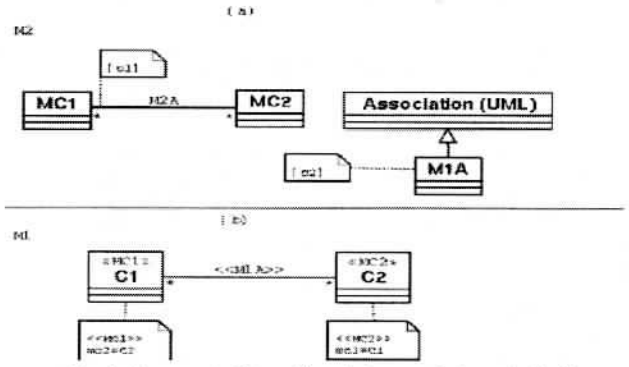

Fig. 3. Representación gráfica de las asociaciones inducidas

En algunos casos lo que se requiere en el metamodelo es inducir clases asociativas en los modelos que lo instanciarán. Por ejemplo, en el metamodelo para los modelos de calidad de productos de software aparecen los conceptos de *Dominio* y *Entorno*. En el metamodelo aparece también el concepto de *Modelo de Calidad*. A nivel *M2* se establece una metaasociación entre las metaclases *Dominio* y *Entorno* (que indica que un determinado dominio de productos puede aplicarse a distintos entornos). Se quiere inducir en todas las instancias de este metamodelo la información siguiente: se obtendrá un modelo de calidad distinto para cada par *<dominio, entorno>* que forme parte de la extensión de la metaasociación anterior. Dicho de otra manera, se quiere inducir una clase asociativa de tipo *ModeloCalidad* que esté ligada a cada asociación que se establezca en *M1* entre un dominio concreto de productos con un entorno. Esta situación nos lleva al concepto de clase asociativa inducida por una metaasociación, análogo al de asociación inducida y que no detallamos por razones de espacio.

5. Creación de un metamodelo para modelos de calidad

La figura 4 presenta el modelo genérico. Las metaasociaciones inducen, todas ellas, asociaciones en los modelos del nivel M1. En la figura se han omitido, por razones de claridad del gráfico, las subclases de *Association* y restricciones que corresponden al aparato formal presentado en la sección anterior. Sobre las clases asociativas, no se han incluído las notas también por razones de espacio. Finalmente, una vez más por motivos de claridad, hemos omitido también las siguientes relaciones de herencia: *Concepto* es una superclase de *Unidad*, *Tipo* y *Escala*; por su parte, *ConceptoRefComp* es una superclase de *Entorno*, *Dominio*, *Artefacto*, *Atributo*, *Cuestión*, *ModeloCalidad* y *Métrica*. La relación de herencia entre *Concepto* y otras metaclases tiene por objeto relacionarlas con *Módulo* a través de la metaasociación *encapsula*, induciéndose en M1 una asociación entre las instancias de esas metaclases y la de *Módulo*. La relación de herencia mencionada no se traslada, pues, al nivel M1. En cambio, la herencia entre *Atributo* y sus dos herederos sí va a trasladarse a M1.

Un aspecto que cabe explicar de este metamodelo es el que se refiere a la forma de representar la asignación de valores a los atributos de calidad. Debemos tener en cuenta que más de un modelo de calidad puede ser aplicable a un mismo artefacto y que varios de estos modelos aplicables pueden contener un mismo atributo. Sin embargo, no podemos asegurar que en todos los casos se va a usar la misma métrica, de manera que un mismo atributo puede ser valorado de forma distinta incluso para un mismo artefacto. Representamos esta situación a través de la metaasociación *aplicable* que inducirá una clase asociativa instancia de *ModeloAplicable* (que representa a los modelos de calidad aplicables a un artefacto *A*) y con otra metaasociación entre esta última clase y *Atributo*, que inducirá otra clase asociativa (ahora instancia de *Medición*) que representa las asignaciones a un atributo del artefacto *A* para cada modelo aplicable a ese artefacto.

Destacamos a continuación los aspectos más relevantes del modelo genérico en esta versión concordante con MOF:

- *Módulo* representa las diferentes clases de módulo. Sus instancias vendrán dadas por las notaciones de representación de la calidad. La misión de *Concepto* la indica su propio nombre. *encapsula* indica que cualquier clase de módulo deber servir para contener algún concepto. Sin embargo, puede haber conceptos que no formen parte explícitamente de un módulo.
- *subcomponenteDe* y *refinadoPor*: puede haber módulos y conceptos que sean no refinables y/o imposibles de descomponer.
- *Dominio* se instanciará con las diferentes visiones existentes para clasificar los dominios de *software*. Análogamente, *Entorno* hace lo propio con los entornos de uso. *usadoEn* los asocia de manera que toda forma de representar los dominios de *software* debe poder aplicarse en combinación con alguna forma de representar los entornos de uso y viceversa. Nuestra concepción del modelo de calidad como algo asociado a un par <dominio, entorno> se refleja de esta manera en el metamodelo.
- En las clases *ModeloCalidad*, *ModeloAplicable* y *Atributo*, cuyo papel se intuye a partir de sus nombre, reside el núcleo del modelo. Nótese que las metaasociaciones *incluye* y *evalúa* fijan para un estilo de modelo de calidad un tipo de atributos. Ello no impide que en un refinamiento concreto del metamodelo la clase instancia de *Atributo* pueda tener especializaciones, como veremos en la sección siguiente.

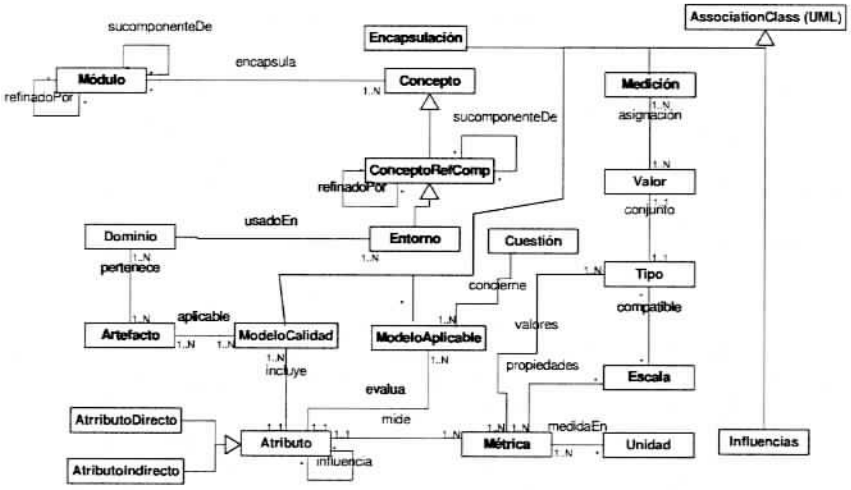

Fig. 4. El modelo genérico

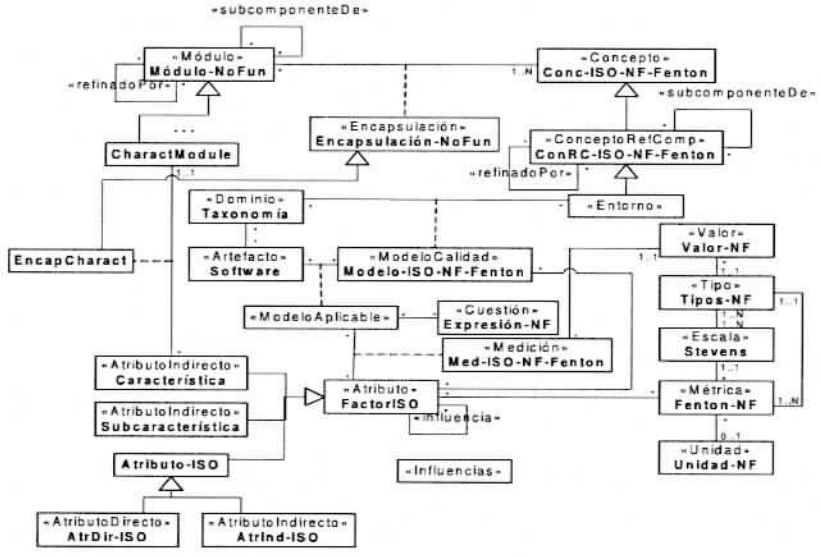

Fig. 5. Un modelo de referencia

- Aunque cada conjunto de valores viene determinado por su tipo y viceversa, se representan por separado con la metaasociación *conjunto* para mayor claridad.

- *Cuestión* representa las maneras de representar requisitos o condiciones (predicados, expresiones booleanas, etc.).

6. Un Modelo de Referencia creado a partir del Metamodelo

Describimos aquí el modelo de referencia (véase figura 5) que hemos usado como base en algunas experiencias. Se obtiene de la combinación del uso del standard ISO/IEC 9126 [ISO01] y el lenguaje NoFun [Fra98], que incorpora las ideas clásicas relativas a métricas y escalas [FP98, Ste46].

Cada clase del modelo de referencia es instancia de una metaclase del modelo genérico, lo que se indica con los estereotipos correspondientes.

Asimismo, las asociaciones (cuyo estereotipo no es estrictamente el que consta en la figura si se tiene en cuenta la definición de asociación inducida) vienen inducidas por las metaasociaciones del modelo genérico.

Es interesante el efecto de las relaciones de herencia del metamodelo en el modelo:

- La herencia que clasifica los atributos en directos e indirectos en el metamodelo (que es trasladada a todos los modelos de referencia) la combinamos en este con la clasificación en características, subcaracterísticas y atributos.
- La herencia entre *Concepto* y *ConceptoRefComp* y el resto de metaclases no se traslada obligatoriamente a los modelos de referencia. En el que nos ocupa, sí se traslada debido a la estructura modular de NoFun. El modelo genérico impone una *Encapsulación* entre cada instancia de cada concepto y la instancia de *Módulo*. En la figura presentamos únicamente la que corresponde a *Característica*.

7. Conclusiones

Las aportaciones más relevantes del presente trabajo son de naturaleza doble: algunas van asociadas a nuestro objetivo inicial (la mejora de [BF03]) y otras a la de [RF02].

- Se usa una metodología de extensión del metamodelo claramente definida, que consta de dos pasos: la extensión en sí y la generación de un *profile*, que permite razonar a un alto nivel (en el metamodelo).
- La formalización de la propuesta da acceso a la realización de herramientas de desarrollo de modelos de referencia y de dominio.
- La relación con el metamodelo de UML proporciona a la propuesta una semántica expresada en términos conocidos por la comunidad. La distribución de modelos en los niveles MOF formaliza y clarifica el grado de abstracción de cada uno y da significado concreto a los pasos de refinamiento (obtención de modelos de referencia a partir del genérico y de modelos de dominio a partir de los de referencia).
- Hemos reformulado algunos detalles de los modelos con mayor precisión e incluso simplicidad (e.g., hemos simplificado la definición de la asignación de valores a un atributo, que anteriormente se realizaba mediante una asociación ternaria).
- La definición completa de un modelo de referencia permite ver más claramente el proceso de refinamiento y las restricciones que le impone el modelo genérico.

- Se ha definido formalmente el concepto de asociación inducida, cuya aplicación puede ser necesaria no sólo en la modelización de la calidad sino en otros ámbitos. Este concepto puede extenderse a herencia inducida con un significado similar. Todo ello abre la puerta a un proceso de mejora de la metodología de [RF02].

Existen similitudes con el marco general presentado por diversos grupos españoles [Ols+02, Gar+04] ya analizado en [BF03] y que no se integra con MOF.

Bibliografía

[BF03] X. Burgués, X. Franch. "Formalización de la calidad del software mediante una estructura jerárquica de modelos". En *JISBD*, Alicante, 2003.

[DAML04] Defense Advanced Research Projects Agency (DARPA). *The DARPA Agent Markup Language Homepage*. Available at http://www.daml.org/. Last accessed June 04.

[Fir03] D. G. Firesmith. "Using Quality Models to Engineer Quality Requirements". *Journal of Object Technology*, 2(5), 2003.

[FP98] N. Fenton, S.L. Pfleeger. Software Metrics: A Rigorous and Practical Approach. PWS.1998

[Fra98] X. Franch. "Systematic Formulation of Non-Functional Characteristics of Software". In *Proceedings of the 3rd IEEE* (ICRE), Colorado Springs (CO, USA), 1998.

[Gar+04] F. García, F. Ruíz, M.F. Bertoa, C. Calero, M. Genero, L. Olsina, M. Martín, C. Quer, N. Tondori, S. Abrahao, A. Vallecillo, M. Piattini. "Una Ontología de la Medición del Software". Informe Técnico UCLM DIAB-04-04-2, Febrero 2004.

[GV95] R. Glass, I. Vessey. "Contemporary Application Domain Taxonomies". *IEEE Sofware*, 12(4), 1995.

[IEE92] IEEE Standard 1061-1992. *Standard for a software quality metrics methodology*. 1992.

[ISO01] ISO/IEC Standard 9126-1 Software Engineering - *Product Quality - Part 1*, 2001.

[KHL01] B. Kitchenham, R. Hugues, S.G. Linkman. "Modeling Software Measurement Data". *IEEE Transactions on Software Engineering*, 27(9), 2001.

[KKP90] S. Keller, L. Kahn, R. Panara. "Specifying Software Quality Requirements with Metrics". *System and Software Requirements Engineering* – IEEE Computer Society, 1990.

[MOF04] *MOF 2.0 Core Specification*. http://www.uml.org/, last accessed June 2004.

[Ols+02] L.A. Olsina, M. F. Bertoa, G.J. Lafuente, M.A. Martín, M. Katrib, A. Vallecillo. "Un Marco Conceptual para la Definición y Explotación de Métricas de Calidad". *JISBD* 2002.

[RCDT02] A. Ruíz, R. Corchuelo, A. Durán, M. Toro. "Automated negotiation of quality rquirements". *VII Jornadas de Ingeniería del Software y Bases de Datos* (JISBD), 2002.

[RF02] J.M. Ribó, X. Franch: "Una Metodología a dos Niveles para Extender el Metamodelo de UML". En *JISBD*, Madrid, Noviembre 2002.

[Ste46] S.S. Stevens. "On the theory of scale types and measurement". *Science* 103, 1946.

[UML04] *UML 2.0 Specifications*. http://www.uml.org/, last accessed June 2004.

[Woh+04] E: Wohlstadter, S. Tai, T. Mikalsen, I. Rouvellou, P. Devanbu. "GlueQoS: Middlware to Sweeten QoS Policy Interactions". IEEE *Procs. 26th ICSE*, Edinburgh (UK), 2004.

[XML04] World Wide Web Consortium (W3C). *Extensible Markup Language (XML)*. Avaiable at http://www.w3.org/XML/. Last accessed June 04.

[Zus98] H. Zuse. *Framework of Software Measurement*. De Gruyter, 1998.