

COSTUME: Un método para la combinación de modelos de calidad

Juan P. Carvallo, Xavier Franch, Gemma Grau, Carme Quer

Universitat Politècnica de Catalunya (UPC)

c/ Jordi Girona 1-3 (Campus Nord, C6) E-08034 Barcelona (Catalunya, España)

{carvallo, franch, ggrau, cquer}@lsi.upc.es

Resumen

El uso de modelos de calidad durante la selección de componentes COTS (Comercial, Off-The-Shelf components) proporciona un entorno adecuado para la descripción de los dominios a los que pertenecen dichos componentes. La selección en este caso resulta más eficiente y fiable, pues tanto las descripciones de los componentes COTS como las de los requisitos de calidad de los usuarios pueden traducirse a los factores de calidad definidos en el modelo. En este artículo tratamos la construcción de modelos de calidad para la Sistemas Software basados en Componentes COTS (CCSS), que definiremos como sistemas compuestos por varios componentes COTS interconectados. Los procesos de selección llevados a cabo para obtener un CCSS no requieren la selección de un único producto COTS, sino de varios. Como consecuencia, en vez de usar un modelo de calidad tradicional, necesitaremos un modelo más elaborado, definido como la composición de todos los modelos de calidad pertenecientes a los componentes COTS que forman el CCSS. Proponemos un método de apoyo para la construcción de modelos de calidad para CCSS basado en la aplicación de cuatro actividades. Nuestro objetivo es conseguir que los factores de calidad que aparecen en los CCSS estén definidos en términos de los factores de calidad de los componente COTS y, de esta forma, obtener eficientemente modelos de calidad que cumplan el estándar de calidad ISO/IEC 9126-1 y sean fáciles de entender, analizar, mantener y reutilizar.

1. Introducción

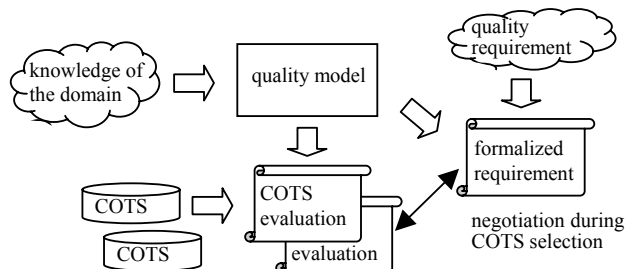
La cantidad de componentes COTS (Commercial Off-The-Shelf components) [CL00] aumenta día a día debido a dos factores fundamentales: el incremento del uso de las tecnologías software basadas en componentes y la creación continua de nuevas vías de comunicación que acercan los proveedores y consumidores de dichos productos. Existe, por lo tanto, una necesidad creciente de identificar y cualificar los tipos de COTS¹ para mejorar la eficiencia y eficacia en su obtención [FSR96]. Los *requerimientos de calidad* han sido reconocidos como cruciales en los métodos y procesos hasta ahora propuestos para llevar a cabo dicha actividad [Kon96, MN98]. Sin embargo, es necesario un gran esfuerzo para

obtener, de forma eficiente, descripciones fiables y comprensivas de la calidad de los COTS.

Los modelos de calidad [ISO86] son una forma especialmente atractiva de estructurar dichas descripciones. Un *modelo de calidad* proporciona una jerarquía de *factores de calidad* del software así como *métricas* para evaluarlas. Los modelos de calidad pueden usarse en muchos contextos, por ejemplo [Dro96, KP96] los usan para evaluar la calidad de los sistemas desarrollados.

En este artículo estamos interesados en el uso de modelos de calidad para facilitar la selección de COTS. En dicho contexto, los modelos de calidad están relacionados con dominios de COTS en vez de estarlo con productos particulares. De esta manera, los modelos de calidad deben abarcar los factores de calidad que son comunes en todos los productos COTS del dominio. Una vez el modelo de calidad esta disponible, tanto las evaluaciones de los factores de calidad de los COTS como los requisitos de calidad que serán usados durante la selección deben traducirse a conceptos definidos en el modelo (ver figura 1).

Figura 1 Uso de los modelos de calidad en la selección de COTS.



En trabajos previos [FC02, FC03], presentamos un método para construir modelos de calidad para dominios de COTS basado en el estándar de calidad ISO/IEC 9126-1 [ISO01]. También exploramos su uso en el contexto de la selección de COTS. En este artículo, extendemos considerablemente el ámbito de dicho trabajo abordando la construcción de modelos de calidad para *sistemas software basados en componentes COTS* (CCSS). Los CCSS son sistemas que están compuestos de varios COTS interconectados. Nuestra propuesta puede ser aplicada sobre tipos particulares de CCSS como las *product lines* [CN02] y los *sistemas de información cooperativa* [ALMN99].

¹ En este artículo, abreviamos “componente COTS” con COTS.

El objetivo de este artículo es identificar y definir las partes que componen un modelo de calidad para CCSS y presentar el método COSTUME para construirlos.

2. Introducción a COSTUME

El punto de partida de esta investigación es el método para construir modelos de calidad explicado en otros artículos [FC02, FC03], centrado en el estándar de calidad ISO/IEC 9126-1 [ISO01], basado en nuestras experiencias académicas y industriales en el campo de la selección de COTS.

El caso más completo que desarrollamos es un modelo de calidad para el dominio de los servidores de correo [CFQ03]. En la construcción de dicho modelo descubrimos algunos defectos resultado de nuestro método y comprendimos la necesidad de mejorarlo considerando de forma explícita el caso de los modelos de calidad para CCSS. El resultado es COSTUME (COmposite SofTware system qUality Model dEvelopment), un método orientado a la definición de modelos de calidad para CCSS que consiste en las cuatro actividades siguientes:

- *Actividad 1. Análisis del dominio y del CCSS.* Identificación de los elementos organizativos que rodean el CCSS, así como todos los sistemas externos con los que el CCSS deberá interactuar.
- *Actividad 2. Descomposición del CCSS en subsistemas.* A partir de la actividad anterior el CCSS es descompuesto en elementos individuales, de manera que cada uno ofrece servicios bien definidos.
- *Actividad 3. Construcción de modelos individuales para el CCSS.* Aplicación de nuestro método [FC02, FC03] sobre cada elemento del CCSS, para construir un modelo de calidad basado en el ISO/IEC 9126-1.
- *Actividad 4. Combinación de los modelos de calidad individuales.* Combinación de los modelos individuales para obtener un único modelo de calidad ISO/IEC 9126-1 para todo el CCSS.

En las secciones siguientes definimos de forma más formal estas actividades y usamos un caso de estudio para presentarlas en detalle.

3. Análisis del entorno

Los CCSS no funcionan de forma aislada, sino que se comunican con otros sistemas y personas que actúan conjuntamente como su *entorno*. La primera actividad en COSTUME es identificar los actores que interactúan con el CCSS en su entorno.

Durante todo el artículo, usaremos un CCSS para servidores de correo como ejemplo. Los sistemas servidores de correo son el núcleo principal de la comunicación y coordinación de las infraestructuras de cualquier compañía. Los sistemas servidores de correo

son un buen caso de estudio por varias razones: sus propiedades estructurales (un amplio rango de funcionalidades, fuertes dependencias con el entorno, etc.), su uso generalizado y la existencia de un gran número de productos relacionados con este dominio.

En la tabla 1 se presenta una propuesta de actores para el entorno de los servidores de correo, en la que identificamos el tipo del actor y también damos una breve descripción de su principal objetivo.

Actor	Abb.	Type	Goal
Mail Server System	MSS	Software	Provide communication infrastructure
Mail Client System	MCS	Software	Provide access to messages
Mail Server User	MSU	Human	Send and get messages
Mail Server Administrator	MSA	Human	Put mail server to work accurately and efficiently
Firewall	Fwll	Hardware	Filter incoming requests

Tabla 1 Actores del entorno de los servidores de correo.

Las relaciones entre estos actores pueden ser modeladas gráficamente utilizando modelos basados en actores. En nuestras experiencias hemos aplicado el enfoque i^* de Yu [Yu97] para el modelado basado en actores. En particular, usamos el modelo de Dependencias Estratégicas (SD) para modelar redes de relaciones de dependencia entre los actores de un sistema. Las oportunidades disponibles para los actores son exploradas emparejando un *dependier*, que es un actor que “quiere”, con un *dependee* que tiene la “habilidad” de dar lo que quiere el *dependier*. Un modelo de dependencias a alto nivel del CCSS puede ser desarrollado encajando las habilidades del *dependee* con las necesidades del *dependier*. El enfoque i^* está especialmente indicado para nuestras necesidades, pues fue propuesto para modelar sistemas socio-técnicos que involucran diferentes actores software, humanos o organizacionales.

La figura 2 muestra el modelo SD i^* para el caso de los servidores de correo. Los objetivos (*goals*) y recursos (*resources*) están relacionados con las funcionalidades principales que ofrece el servidor de correo: mailing, facilidades de cooperación, administración de los listados de direcciones y tareas de administración. Los objetivos no funcionales (*soft goals*) son identificados siguiendo los conceptos del ISO/IEC. Encontramos, por ejemplo, objetivos no funcionales referentes a la seguridad y la eficiencia. Como sólo estamos interesados en la parte del entorno que involucra el CCSS que se analiza, en nuestros modelos de calidad no representamos las dependencias entre los actores y su entorno. Remarcamos que el comportamiento del servidor de correo también puede depender de los actores del entorno, como se refleja en las dos dependencias establecidas desde el MSS hasta el MSA.

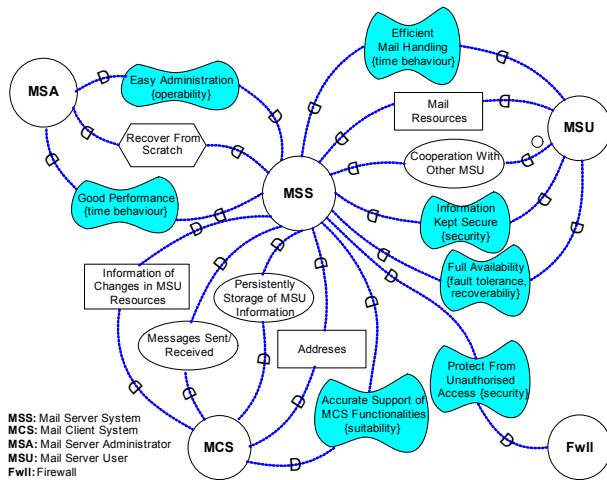


Figura 2 Modelo de entorno para los servidores de correo.

4. Descomponiendo el CCSS en subsistemas

En COSTUME debe descomponerse adecuadamente el CCSS en componentes individuales para facilitar su análisis separado. A pesar de esto, no partimos de una visión basada en componentes sino de una basada en actores: identificamos los actores que participan en el CCSS y establecemos relaciones entre ellos. Como resultado, descomponemos el sistema en sus partes esenciales, destacando los servicios proporcionados por estos actores en vez de la disposición física del sistema en los componentes.

Esta actividad se realiza siguiendo dos caminos concurrentes:

- Orientación al mercado. Es necesario tener un buen conocimiento de los tipos de herramientas actualmente disponibles en el mercado y las funcionalidades que proporcionan [BTV02]. El continuo análisis de *white reports* y documentos técnicos de consultorías profesionales son un factor clave para el éxito.
- Orientación a objetivos. Es necesario tener la habilidad de identificar los actores del sistema y sus objetivos (cada actor tiene asignado un objetivo principal que debe conseguir). Técnicas existentes para el análisis e identificación de objetivos [Ant96] pueden ser adaptadas a nuestras necesidades específicas.

En COSTUME, las dependencias del modelo de entorno conducen a la identificación de algunas categorías de actores. Por ejemplo, la dependencia del objetivo *Cooperation with other MSU* hace surgir la necesidad de actores orientados al *groupware* y nuestro conocimiento del mercado revela la existencia de herramientas de *meeting scheduling*, y de transmisión de voz y de videoconferencia. En la tabla 2 mostramos los actores identificados agrupados por categoría (omitiendo los objetivos de los actores por problemas de espacio). La

tercera columna muestra las dependencias del modelo de entorno (ya mostradas en la figura 2) que justifican la necesidad del actor.

5. Construcción de modelos individuales

En [FC02, FC03] propusimos un método para construir modelos de calidad para dominios independientes de COTS que cumplan la ISO/IEC. Dicho método consta de una serie de pasos para adaptar un modelo de calidad inicial basado en el estándar ISO/IEC 9126-1 [ISO01] al dominio en cuestión. Partiendo de las seis características de calidad y su descomposición en subcaracterísticas:

- Añadimos nuevas subcaracterísticas específicas del dominio, refinar la definición de algunas existentes o bien eliminar algunas.
- Creamos una jerarquía de subcaracterísticas de calidad.
- Descomponemos las subcaracterísticas en atributos de calidad, de manera que representen factores particulares y observables en los COTS pertenecientes al dominio.
- Descomponemos atributos complejos en atributos simples directamente medibles.
- Determinamos las métricas para los atributos.

El resultado de este método es un modelo de calidad individual.

Definición 1. Modelo de calidad individual.

Un *modelo de calidad individual* para un dominio software D , es un modelo de calidad QM_D que cumple la ISO/IEC 9126.

La construcción de estos modelos de calidad individuales está expuesta a muchos factores de riesgo. A continuación, exponemos algunas medidas aplicables a las dificultades habituales. En primer lugar y con la intención de proporcionar un marco común basado en el estándar, mencionamos el uso de ontologías para la calidad del software [FGD02]; su uso facilita no sólo la construcción de los modelos individuales, sino también su futura combinación en la siguiente actividad de COSTUME. En segundo lugar proponemos la reutilización pues, a pesar de que la situación en que se construyen los modelos de calidad es diversa, los modelos de calidad construidos para otras experiencias pasadas (p.e., usados en otros tipos de CCSS) pueden ser reutilizados. El resto de modelos deberán ser construidos para el proyecto particular pero podrán quedar incompletos pues tan sólo tendrán en cuenta los requisitos de dicho proyecto. Volveremos a estas cuestiones en la sección 7.

Category	Actors	Contexto de aplicación
Communication Support	Mail Servers	• Messages Sent/Received
	Routing Tools	• Efficient Mail Handling
Groupware Support	Meeting Scheduler Tools	• Co-operation With Other MSU
	Voice and Video-Conference Tools	
	Chatting Tools	
	Instant Messaging Tools	
	News Servers	
Resources	Lists Servers	
	Directory Services	• Mail Resources • Addresses • Persistent Storage of MSU information
Security Support	Compression Tools	• Mail Resources
	Anti-Spam Filter Managers	• Information Kept Secure
Administrative Support	Anti-virus Tools	
	Backup and Recovery Tools	• Full Availability
	Message Tracking Tools	• Efficient Mail Handling • Good Performance
	Configuration and Administration Tools	• Easy administration

Tabla 2 Actores para el caso de los servidores de correo

6. Composición de Modelos de Calidad

Los modelos de calidad individuales dan una visión individual, aislada e incompleta de las partes del CCSS. Para conseguir una visión integrada de la calidad de todo el CCSS, la siguiente actividad consistirá en componer estos modelos de calidad. La composición se basa en combinar de forma apropiada los factores de calidad de los modelos de calidad. De manera formal:

Definición 2. Modelo de calidad de CCSS.

Sea S un CCSS y A el conjunto de actores en que se descompone S . El modelo de calidad de CCSS para S está definido como una tupla $QM_S = (\{QM_D\}_{D \in A}, QM_S, Map_S)$:

- $\{QM_D\}_{D \in A}$ son modelos individuales que consideran los actores de A como dominios. Los llamaremos *modelos de calidad componentes*.
- QM_S es un modelo de calidad que cumple la ISO/IEC 9126-1 y que llamaremos *modelo de calidad compuesto*.
- Map_S es una familia de 2 funciones, $Map_S = (MapSubcars_S, MapAttrs_S)$, que asigna algunas de las subcaracterísticas respecto a los atributos de QM_S a aquellos pertenecientes a $\{QM_D\}$.

Las características no son tenidas en cuenta de forma explícita porque se considera que todo modelo de calidad compuesto incluye las definidas en la ISO/IEC 9126-1.

El factor clave de la composición se basa en que la mayoría de los elementos del modelo de calidad están definidos como resultado de relacionar los nuevos elementos con los ya existentes, de manera que sólo unos pocos elementos (por ejemplo, aquéllos que se refieren a conceptos arquitectónicos) aparecen. Por otro lado, la

relación se define mediante la aplicación repetida de ciertos patrones de combinación de subcaracterísticas y atributos que serán definidos en el resto de esta sección.

6.1 Patrones de combinación para subcaracterísticas

Las subcaracterísticas son factores de calidad de alto nivel y, como tales, deberían estar definidas en la ontología propuesta en la sección anterior. Eso sí, nuestro enfoque no está restringido a ninguna ontología en particular, la ontología puede incluso no existir, pero proponemos su existencia como una herramienta conceptual útil que facilita la construcción de las partes del modelo del CCSS. En particular, si los modelos del componente comparten muchas de sus características, la aplicación de los patrones se convierte en una tarea más simple.

Sea S un CCSS, QM_S su modelo de calidad compuesto y $QM_1, QM_2 \in \{QM_D\}$ dos modelos de calidad componentes. Podemos identificar 4 patrones para combinar subcaracterísticas (ver figura 3).

Para simplificar, las definiciones son establecidas sobre pares de elementos, fácilmente generalizables a n elementos. Cuando es necesario se usan prefijos representados por “:”. Utilizamos la función *predecesor* para obtener el predecesor del factor de calidad en la jerarquía. La combinación de patrones debe ser aplicada de arriba a abajo (*top-down*), de manera que primero se aplica para construir los niveles superiores de la jerarquía y luego se completan los niveles inferiores.

Patrón de identificación

- Precondiciones:

- 1) x e y son dos subcaracterísticas, $x \in QM_1, y \in QM_2$.
- 2) existe una subcaracterística $p \in QM_S$ de manera que $MapSubcars_S(p) = \{\text{predecesor}(x), \text{predecesor}(y)\}$.
- 3) $z \notin QM_S$.

- Patrón de aplicación: Identificación(x, y, z)

- Postcondiciones:

- 1) $z \in QM_S$ de manera que $\text{predecesor}(z) = p$.
- 2) $MapSubcars_S(z) = \{QM_1::x, QM_2::y\}$.

Pattern	QM_1	QM_2	QM_S
Identification			
Nesting			
Abstraction		N/A	
System	N/A	N/A	

Figura 3 Patrones para las subcaracterísticas

Contexto de Aplicación

Hay muchas subcaracterísticas que aparecen de forma repetida en un gran número de modelos de calidad. En este caso, simplemente se añade la subcaracterística al modelo compuesto. De manera general, el patrón de identificación permite diferentes nombres en los diferentes modelos, aunque el caso habitual en este patrón es que tengan el mismo nombre, sobretodo si se está usando una ontología.

Ejemplo

Un caso básico son las subcaracterísticas de la ISO/IEC 9126-1. Además, en nuestro caso de estudio, tanto los modelos de calidad de las herramientas de voz y videoconferencias como los de las herramientas de chat, descomponen su subcaracterística *Security* en *Local Security* y *External Security*. Hemos aplicado el patrón de identificación para obtener las mismas subcaracterísticas en el modelo compuesto.

Patrón de anidamiento

Precondiciones:

1) x e y son dos subcaracterísticas, $x \in QM_1$, $y \in QM_2$. En este caso, QM_1 y QM_2 pueden ser iguales.

2) Existe una subcaracterística $p \in QM_S$ de manera que $MapSubcars_S(p) = \{\text{predecesor}(x), \text{predecesor}(y)\}$.

3) $x, y, z \notin QM_S$.

▪ Patrón de aplicación: Anidamiento(x, y, z)

Postcondiciones:

1) $x, y, z \in QM_S$ de manera que $\text{predecesor}(z) = p$, $\text{predecesor}(x) = \text{predecesor}(y) = z$.

2) $MapSubcars_S(x) = \{QM_1::x\}$, $MapSubcars_S(y) = \{QM_2::y\}$.

3) $z \notin \text{dom}(MapSubcars_S)$

Contexto de Aplicación

Al contrario de lo que sucede con la identificación, muchas subcaracterísticas deben mantenerse separadas en el modelo compuesto porque representan distintos factores de calidad. A pesar de esto, no es suficiente añadirlas tal cual porque se empobrecería el modelo resultante. Por eso identificamos una nueva subcaracterística (z en la definición precedente) que ayudará a estructurar el modelo compuesto. La nueva subcaracterística puede crearse siguiendo dos criterios diferentes:

- Orientado a dominio. La nueva subcaracterística z junta dos subcaracterísticas x e y del mismo dominio (i.e., QM_1 y QM_2 son el mismo).
- Orientado a funcionalidad. La nueva subcaracterística z junta dos subcaracterísticas x e y de dos dominios diferentes (i.e., QM_1 y QM_2 son diferentes) que hacen referencia a una misma funcionalidad. En este caso la nueva subcaracterística representa un nuevo concepto y la ontología debe ser actualizada.

Ejemplo

Una aplicación del patrón orientada al dominio aparece cuando se considera las subcaracterísticas de *Accuracy: Accurate Scanning&Repair* y *Actualization of Lists* del dominio de los antivirus. En el modelo compuesto se agruparán introduciendo una nueva subcaracterística que llamaremos *Antivirus Accuracy*, definida como sucesora de *Accuracy*.

Un caso de orientación a la funcionalidad aparece cuando se consideran las subcaracterísticas de *Security: User Privileges* del dominio de los servicios de directorio y *Password Management* de las herramientas de copia y recuperación. Las hemos combinado en el modelo compuesto mediante la definición de la subcaracterística predecesora *Internal Security*.

Patrón de abstracción

Precondiciones:

1) x es una subcaracterística, $x \in QM_1$.

2) existe una subcaracterística $p \in QM_S$ de manera que $MapSubcars_S(p) = \{\text{predecesor}(x)\}$.

3) $x, z \notin QM_S$.

▪ Patrón de aplicación: Abstracción(x, z)

Poscondiciones:

1) $x, z \in QM_S$ de manera que $\text{predecesor}(z) = p$, $\text{predecesor}(x) = z$.

2) $MapSubcars_S(x) = \{QM_1::x\}$.

3) $z \notin \text{dom}(MapSubcars_S)$

Contexto de Aplicación

Este es un caso especial del patrón de anidamiento. Generalmente aparece en dominios pequeños en los que las subcaracterísticas de alto nivel se descomponen directamente en atributos. Cuando se incorporan estas subcaracterísticas en el modelo de calidad compuesto, debido a que éste ya tiene una descomposición en subcaracterísticas en el mismo nivel, resulta conveniente crear una nueva subcaracterística que agrupe estos atributos y así obtener un modelo más equilibrado.

Ejemplo

La subcaracterística ISO/IEC 9126-1 *Fault Tolerance* no esta descompuesta en el dominio de las herramientas de *routing* de manera que atributos como *Time to Recalculate Path* son sucesores de esta subcaracterística. Sin embargo, al construir el modelo compuesto, es conveniente añadir una nueva subcaracterística *Adaptable Routing* que sea sucesora de *Fault Tolerance* y que agrupe los atributos mencionados.

Patrón de Sistema

Precondiciones:

1) x es una subcaracterística, $x \in QM_1$.

▪ Patrón de aplicación: Sistema(x)

Poscondiciones:

1) "Sistema(x)" $\in QM_S$ tal que $\text{predecesor}(\text{"Sistema}(x)") = x$.

2) "Sistema(x)" $\notin \text{dom}(MapSubcars_S)$

Contexto de Aplicación

Este patrón permite introducir una subcaracterística para agrupar todos aquellos atributos que no pueden ser definidos en términos de los factores de calidad de los componentes. Este tipo de atributos acostumbran a surgir de la arquitectura del sistema.

Ejemplo

En el modelo de calidad compuesto suele aparecer la nueva subcaracterística *System Interoperability* para agrupar los atributos que evalúan la capacidad de interoperabilidad que tienen los componentes de los dominios del CCSS entre ellos.

La figura 4 muestra un ejemplo final de la parte de *suitability* que usa estos cuatro patrones. El patrón de identificación es usado para identificar las subcaracterísticas de *Suitability* de los diferentes dominios. El patrón de anidamiento orientado a dominio es usado para conservar la parte de la *suitability* inherente en los servidores de correo (p.e., *message management*) y los *meeting schedulers* (p.e., *meeting arrangement* y *calendar*). El patrón de anidamiento orientado a funcionalidad se aplica para agrupar funcionalidades relacionadas como *folders* y *meeting management* (en ambos casos, habrá atributos relativos a la gestión de grupos de ítems, ya sea mensajes o reuniones). El patrón de abstracción introduce nuevas subcaracterísticas para poner la *suitability* de los algoritmos de compresión al mismo nivel que los otros. Finalmente, el patrón de sistema es utilizado para introducir una nueva subcaracterística que puede agrupar futuros atributos.

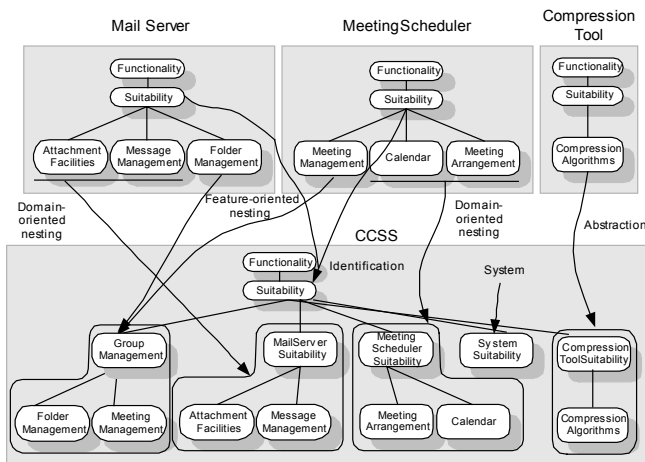


Figura 4 Ejemplo de aplicación de los patrones para subcaracterísticas

En versiones previas de COSTUME estudiamos otros patrones para las subcaracterísticas pero su uso no fue necesario en nuestro caso de estudio y los descartamos.

Los patrones definidos en esta sección pueden ser aplicados de diferentes maneras. Hemos encontrado que

la estrategia que mejor funciona en la mayoría de casos puede definirse como:

- El patrón de identificación es usado en los niveles superiores de la jerarquía del modelo.
- El patrón de anidamiento orientado a dominio suele aplicarse en los niveles inferiores de la jerarquía de subcaracterísticas. Ocasionalmente en este nivel también se aplican algunos patrones de identificación cuando se combinan atributos de dominios parecidos (e.g. *antivirus* y *antispam*).
- El patrón de abstracción es usado para nivelar la jerarquía.
- El patrón de sistema sólo se aplica en el primer nivel de las subcaracterísticas para evitar proliferación.
- El patrón de anidamiento orientado a funcionalidad sólo se usa cuando se identifica un nuevo concepto en el modelo compuesto.

Si se sigue esta estrategia, la aplicación de patrones se convierte en un proceso mecánico a excepción del último caso, que en nuestras experiencias se aplica menos de un 5%.

6.2 Patrones de combinación para atributos

Los atributos de calidad para el modelo compuesto para el CCSS son básicamente de tres tipos: atributos heredados de los modelos individuales, atributos redefinidos a partir de los conceptos ontológicos de los modelos y nuevos atributos que representa nuevas funcionalidades. Los patrones de atributos tienen en cuenta estos tres casos.

Sea QM_S el modelo de calidad compuesto y $QM_1, QM_2 \in \{QM_D\}$ dos modelos de calidad individuales del CCSS, identificamos 5 patrones de combinación de atributos. Como en el caso de las subcaracterísticas, las definiciones se hacen respecto a pares de atributos pero pueden extenderse al caso general.

Patrón de delegación

- Precondiciones:
 - 1) x es un atributo, $x \in QM_1, x \in QM_2$.
 - 2) existe una subcaracterística $p \in QM_S$ tal que $MapSubcars_S(p) \supseteq \{predecesor(x)\}$.
- Patrón de aplicación: Delegación(x, z)
- Postcondiciones:
 - 1) $x \in QM_S$ de manera que $predecesor(x) = p$.
 - 2) $MapAttrs_S(x) = \{QM_1::x, QM_2::x\}$.
 - 3) Las métricas de x se definen como $x = QM_1::x$.

Contexto de aplicación

Dado un atributo en particular, se da el caso que en uno de los componentes este atributo predomina sobre el resto. Por lo tanto, el atributo del modelo compuesto se definirá sólo en términos del predominante.

Ejemplo

El atributo *Existence of Log Files* puede estar en varios componentes del CCSS, pero estamos únicamente

interesados en saber si está presente el servidor de correo. En este caso consideramos que el CCSS genera *Log Files* si el servidor de correo los genera.

Patrón de Redefinición

- Precondiciones:
 - 1) x es un atributo, $x \in QM_1, x \in QM_2$.
 - 2) existe una subcaracterística $p \in QM_S$ tal que $MapSubcars_S(p) \supseteq \{\text{predecesor}(x)\}$.
- Patrón de aplicación: Redefinición(x, z)
- Postcondiciones:
 - 1) $x \in QM_S$ de manera que $\text{predecesor}(x) = p$.
 - 2) $MapAttrs_S(x) = \{QM_1::x, QM_2::x\}$.
 - 3) Las métricas de x se definen como $x = K$.

Contexto de aplicación

Este patrón se aplica cuando el concepto aparece en algún modelo individual y es válido en el modelo compuesto, de manera que su definición del modelo compuesto no depende de los modelos individuales.

Ejemplo

Time in the Market es un atributo de la subcaracterística *Maturity*. Todos los modelos individuales tienen este atributo y el modelo compuesto también deberá tenerlo, pero su valor no será calculado a partir de los otros.

Patrón de Combinación

- Precondiciones:
 - 1) x es un atributo, $x \in QM_1, x \in QM_2$.
 - 2) existe una subcaracterística $p \in QM_S$ tal que $MapSubcars_S(p) \supseteq \{\text{predecesor}(x)\}$.
- Patrón de aplicación: Combinación(x, p)
- Postcondiciones:
 - 1) $x \in QM_S$ de manera que $\text{predecesor}(x) = p$.
 - 2) $MapAttrs_S(x) = \{QM_1::x, QM_2::x\}$.
 - 3) Las métricas de x se definen como $x = f(QM_1::x, QM_2::x)$.

Contexto de aplicación

El caso más habitual de aplicación ocurre cuando se tienen en cuenta todos los atributos de los modelos individuales y se definen métricas para combinarlos.

Ejemplos

Existen varios ejemplos con diferentes funciones de combinación:

- Aditivas. El atributo *total average time to send a message* perteneciente a la subcaracterística *Time Behaviour*, puede definirse como la suma de cinco atributos: *average sending time* (servidores de correo), *virus scanning* (antivirus), *mail compression* (compresión de datos), *time to find destination node* (herramientas de routing), and *time to update log file* (servidores de correo).
- Valor mínimo o máximo. El atributo *Mean Time Between Failures* puede ser definido cómo el mínimo de

los *Mean Time Between Failures* de los modelos individuales.

- Unión e Intersección de conjuntos. El atributo *Languages Of Documentation* de la subcaracterística *Understandability* puede definirse cómo la intersección de los valores de los modelos individuales.
- Otras funciones complejas. La subcaracterística *Communication Protocol* está definida en los modelos individuales cómo una función respecto a una variable. En el modelo compuesto estará definida respecto a dos variables definidas como:

$$\text{CommProt}(x, y) = x::\text{CommProt}(y)$$

donde x e y son dos modelos individuales de componentes (p.e., servidores de correo y servicio de directorio) mientras que $x::\text{CommProt}(y)$ es el protocolo de comunicación que usa x para comunicarse con y .

Patrón de nuevo atributo combinado

- Precondiciones:
 - 1) x es un atributo, $x \in QM_1, x \in QM_2$.
 - 2) existe una subcaracterística $p \in QM_S$.
- Patrón de aplicación: NuevoAtributoCombinado(x, p)
- Postcondiciones:
 - 1) $x \in QM_S$ de manera que $\text{predecesor}(x) = p$.
 - 2) $x \in MapAttrs_S(x)$.
 - 3) Las métricas de x se definen como $x = f(QM_1::x, QM_2::y)$.

Contexto de aplicación

En el modelo compuesto, algunas veces es necesario definir nuevos atributos derivados en función de los existentes. En este caso, el nuevo atributo no se relaciona con los otros pero se usa una métrica para indicar su dependencia. La ontología debe ser actualizada si se considera que el nuevo atributo representa un atributo de calidad importante.

Ejemplo

El atributo *Operating Systems Supported* de la subcaracterística *Portability* puede tener como valor un conjunto de etiquetas. Usar una intersección de estos conjuntos es una definición muy restrictiva debido a que el CCSS del servidor de correo puede instalarse en plataformas distintas. Por esto definimos el atributo *Feasible Operating Systems Combination* que será un conjunto de conjuntos de etiquetas que registrarán las combinaciones válidas de sistemas operativos para los componentes.

Patrón de nuevo atributo de sistema

- Precondiciones:
 - 1) x es un atributo, $x \in QM_1, x \in QM_2$.
 - 2) existe una subcaracterística $p \in QM_S$.
- Patrón de aplicación: NuevoAtributoSistema(x, p)
- Postcondiciones:
 - 1) $x \in QM_S$ de manera que $\text{predecesor}(x) = p$.

2) $x \notin \text{MapAttr}_s(x)$.

3) Las métricas de x se definen como $x = K$.

Contexto de aplicación

Este último caso aparece cuando un atributo no está relacionado con todos los factores de calidad de los modelos individuales. La mayoría de las veces el patrón puede aplicarse para capturar las propiedades arquitectónicas del CCSS pues, tal y como hemos mencionado anteriormente, muchas de las nuevas subcaracterísticas que se crean en el modelo compuesto provienen de la aplicación del patrón de sistema.

Ejemplo

El atributo *cohesion* de la subcaracterística *Maintainability* no depende de los factores de calidad del modelo, sino de la forma en que los COTS se interconectan en el CCSS.

7. Análisis de COSTUME

A primera vista, la construcción de tantos modelos de calidad individuales puede parecer una actividad costosa en tiempo y recursos, pero en esta sección defenderemos su uso. Por un lado, la futura reutilización de los modelos compensa la inversión inicial. Por el otro, su construcción puede ser dirigida por los requisitos permitiendo que algunos modelos de calidad individuales queden incompletos. Finalmente, la estructura interna de los modelos CCSS es adecuada para soportar el inevitable mantenimiento del modelo, que viene de los continuos cambios en las demandas del mercado y la sociedad.

7.1 Construcción dirigida por los requisitos

Debido a las presiones de tiempo a que se someten los proyectos reales, es inviable aspirar a realizar un desarrollo completo de los modelos de calidad individuales. En vez de esto, proponemos su construcción dirigida por los requisitos que concentra el esfuerzo en los factores de calidad directamente relacionados con el proyecto de selección para el que se construye el modelo. Consecuentemente, consideramos un proceso de selección iterativo [MN98, BEFPQ02], en el que el proceso de establecimiento de requerimientos y su evaluación en el modelo de calidad pueden ser intercalados.

No sólo el tipo de requerimiento, sino también su nivel de detalle debe ser considerado. Siguiendo esta pauta, los requisitos de los COTS que no proporcionan las funcionalidades principales del CCSS no deben ser tan detalladas. Por ejemplo, en el caso de las herramientas antivirus, la definición de los requerimientos es abstracta: "The messages sent by the system shall be not infected by virus" sin especificar, p.e. que acciones deben realizarse cuando un virus es detectado. Consecuentemente, el modelo de calidad para el dominio de los antivirus puede verse reducido a unos pocos atributos.

7.2 Reutilización de Modelos de Calidad

Sin embargo, la visión orientada a requisitos puede ser perjudicial para otros propósitos, pues no estamos pensando en modelos de calidad de usar y tirar, únicamente construidos para el proyecto en el que surgen. Al contrario, pensamos en modelos de calidad que, de manera parcial o total, puedan ser usados en muchos proyectos. De forma específica, la reutilización puede aplicarse en las situaciones siguientes:

- Reutilización en diferentes procesos de selección del mismo tipo de CCSS. A pesar de que los requisitos pueden ser diferentes, es posible ampliar el modelo de calidad del CCSS con nuevos factores de calidad que los cubran.
- Reutilización en la construcción de nuevos modelos para CCSS. Este caso es especialmente cierto en el caso de los modelos de calidad de propósito general como los antivirus y las herramientas de copia y recuperación: una vez construido el modelo individual, puede ser reutilizado para un gran número de modelos de calidad de CCSS.
- Finalmente, algunos resultados del estudio del entorno pueden ser reutilizados para la construcción de otros modelos de calidad compuestos en los que aparece el mismo grupo de actores. Por ejemplo, en caso de construir un nuevo modelo compuesto para un sistema de clientes de mail, sus dependencias respecto a los sistemas servidores de correo, pueden ser reutilizadas.

Queda implícito que asumimos la existencia de un repositorio de modelos de calidad que pueden ser usados en los proyectos de selección y como punto de partida para la construcción de nuevos modelos de calidad. También asumimos que este repositorio debe ser actualizado con nuevos modelos de calidad. Proponemos en [CFGQT04] la organización de este repositorio como una taxonomía en la que dominios relacionados pueden compartir partes comunes de sus modelos de calidad. De hecho, considerando la reutilización, se podría redefinir la actividad 3 de COSTUME (construcción de los modelos de calidad individuales de cada dominio) teniendo en cuenta los pasos siguientes:

Paso 1: Localización. Búsqueda en el repositorio de un modelo individual para el dominio. La organización taxonómica del repositorio facilita esta búsqueda y, en caso de no encontrar el modelo, permite acceder a los dominios más próximos y usarlos como punto de partida. Si no existe ningún modelo próximo, se debe usar el modelo ISO/IEC 9126-1 como punto de partida.

Paso 2: Personalización. Ampliación del modelo de calidad de partida teniendo en cuenta los factores de calidad específicos del dominio y los requerimientos del proyecto. Esta ampliación se realiza aplicando el método de construcción de modelos de calidad individuales de la sección 5 dirigido por los requisitos. Sin embargo, sólo

los factores de calidad relacionados con los requerimientos del CCSS serán añadidos. Por otra parte, los factores de calidad ya existentes que no sean considerados en la selección no serán usados en el proceso de selección.

Paso 3: Reorganización. Para facilitar la reutilización futura, una vez los modelos de calidad individuales están contruidos, la taxonomía debe ser reorganizada. Las acciones a realizar pueden ser diversas: dividir nodos de la taxonomía, crear nuevos nodos, eliminar factores de calidad demasiado dependientes del proyecto e incluso completar el modelo de calidad más allá de los requerimientos del proyecto.

7.3 Mantenimiento de modelos de calidad CCSS

El mercado de COTS se ve continuamente afectado por los cambios. Nuevos productos y versiones de los productos existentes aparecen cada pocos meses, incorporando nuevas funcionalidades y mejorando las existentes. Los nuevos factores de calidad que aparecen en cada versión deben ser incorporados en los modelos de calidad.

La estructura interna de los modelos de calidad facilita la identificación y clasificación de nuevos factores de calidad. Adicionalmente a esta propiedad, común a cualquier enfoque basado en modelos de calidad, los modelos de calidad para los CCSS que proponemos son especialmente adecuados para añadir nuevos grupos de funcionalidades mediante la definición de actores y la actualización de sus mapeos. Si analizamos la historia de los servidores de correo, puede comprobarse que los primeros sistemas no proporcionaban algunas funcionalidades como el *meeting scheduling* o los antivirus. Para un futuro próximo, pueden preverse nuevas funcionalidades como el reconocimiento de voz (para dictar los correos) y *domain-ontology alignment* (p.e., para organizar las carpetas o determinar el tema del correo). Los nuevos actores generarán nuevos modelos de calidad individuales que no interferirán con los actuales. Los CCSS se actualizarán teniendo en cuenta estos nuevos componentes.

8. Conclusiones y trabajo futuro

En este artículo proponemos un método para crear modelos de calidad para CCS. El método COSTUME está organizado en cuatro actividades que incluyen: análisis del entorno, descomposición del CCSS, construcción de modelos de calidad para componentes y combinación de modelos de calidad para obtener el modelo de interés. Los factores de calidad del modelo de calidad compuesto serán definidos en términos de los modelos individuales relacionados con los actores del CCSS.

Pensamos que las contribuciones de nuestro trabajo son:

- La definición de un método más preciso para la construcción de modelos de calidad complejos. A pesar de que ya hay unos cuantos métodos para construir modelos de calidad [Dro96, KP96], incluyendo el nuestro [FC02, FC03], éstos no están especialmente orientados al tipo de sistemas compuestos que hemos tratado en este artículo. La precisión citada se basa en la aplicación de una colección de patrones como guía para la construcción del modelo. En el caso de las subcaracterísticas, los patrones son aplicados casi de forma automática. En el caso de los atributos, los atributos proporcionan un marco muy útil para clasificar los nuevos atributos que aparecen en el modelo de calidad.
- El proceso para construir modelos de calidad es eficiente en el sentido que facilita la reutilización. En la sección 7 hemos presentado 3 tipos de reutilización que hacen que nuestro método sea menos costoso que otros. Así mismo, nuestro enfoque dirigido por los requisitos evita malgastar recursos en aquellas partes del modelo que no interesan en el proceso de selección actual.
- La evolución del mercado requiere cambios en los modelos de calidad, pero la separación de los conceptos dentro del modelo de calidad evita modificaciones innecesarias de los modelos de calidad existentes. Por otro lado, los modelos CCSS son altamente trazables: su definición mantiene constancia de los factores de calidad del COTS que afectan los factores de calidad del CCSS. La trazabilidad es una propiedad que facilita el mantenimiento.
- Nuestra definición de los modelos de calidad para los CCSS es dual porque su forma interna es muy estructurada (permite identificar los modelos individuales) y, a la vez, cumple la ISO/IEC 9126-1, hecho que puede posibilitar su mayor aceptación.

Actualmente estamos desarrollando una herramienta para soportar COSTUME. Tenemos un primer prototipo para la construcción de modelos de calidad individuales que tiene como objetivo a corto plazo dar soporte a la aplicación de patrones. Como trabajo futuro, planeamos integrar el concepto de ontología en el método y proponer una ontología propia.

References

- [ALMN99] D. Avison, F. Lau, M. Myers, P.A. Nielsen. "Action Research". CACM 42(1), 1999.
- [Ant96] Annie I. Antón. "Goal-Based Requirements Analysis". In *Procs. 2nd IEEE ICRE* 1996.
- [BEFPQ02] X. Bugués, C. Estay, X. Franch, J.A. Pastor, C. Quer. "Combined Selection of COTS Components". *Procs. 1st ICCBSS, LNCS*, 2002.

- [BTV02] M. Bertoa, J.M. Troya, A. Vallecillo. "A Survey on the Quality Information Provided by Software Component Vendors". 7th ECOOP QAOOSE workshop, 2002.
- [CFGQT04] J.P. Carvallo, X. Franch, G. Grau, C. Quer, M. Torchiano. "Characterization of a Taxonomy for Business Applications and the Relationships among them". In *Procs. 3rd ICCBSS*, LNCS, 2004 (to appear).
- [CFQ03] J.P. Carvallo, X. Franch, C. Quer. "Defining a Quality Model for Mail Servers". *Procs. 2nd ICCBSS*, LNCS, 2003.
- [CL00] D. Carney, F. Long. "What Do You Mean by COTS? Finally a Useful Answer". *IEEE Software*, 17(2), March 2000.
- [CN02] P. Clements, L. Northrop. *Software Product Lines: Practices and Patterns*. SEI series in SE, Addison-Wesley 2002.
- [Dro96] R.G. Dromey. "Cornering the Chimera". *IEEE Software*, 13(1), 1996.
- [FC02] X. Franch, J.P. Carvallo. "A Quality-Model-Based Approach for Describing and Evaluating Software Packages". *Procs. 10th IEEE Joint International Conference on Requirements Engineering (RE)*, 2002.
- [FC03] X. Franch, J.P. Carvallo. "Using Quality Models in Software Package Selection". *IEEE Software*, 20(1), 2003.
- [FGD02] R.A. Falbo, G. Guizzardi, K.C. Duarte. "An Ontological Approach to Domain Engineering". *Procs. 14th SEKE*, 2002.
- [FSR96] A. Finkelstein, G. Spanoudakis, M. Ryan. "Software Package Requirements and Procurement". *Procs. 8th IEEE IWSSD*, 1996.
- [ISO86] *ISO Standard 8402: Quality management and quality assurance-Vocabulary*, 1986.
- [ISO01] *ISO/IEC Standard 9126-1 Software Engineering – Product Quality – Part 1: Quality Model*, 2001.
- [KHL01] B. Kitchenham, R. Hugues, S.G. Linkman. "Modeling Software Measurement Data". *IEEE TSE* 27(9), 2001.
- [Kon96] J. Kontyo. "A Case Study in Applying a Systematic Method for COTS Selection". *Procs. 18th IEEE ICSE*, 1996.
- [KP96] B. Kitchenham, S.L. Pfleeger. "Software Quality: the Elusive Target". *IEEE Software*, 13(1), 1996.
- [MN98] N. Maiden, C. Ncube. "Acquiring Requirements for COTS Selection". *IEEE Software* 15(2), 1998.
- [PS98] M. Papazoglou, G. Schlageter (eds.). *Cooperative Information Systems: Trends & Directions*. Academic Press, 1998.
- [Yu97] E. Yu. "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering". In *Procs. 3rd IEEE ISRE*, 1997.