Ingeniería de Requisitos Web Orientada a Aspectos con Transformación de Modelos

Martín Villalba[1], Juan Eduardo Durán[1]

[1] Facultad de Matemática, Astronomía y Física - Universidad Nacional de Córdoba - Medina Allende S/N - 5010 Córdoba - Argentina
{villalba,duran}@famaf.unc.edu.ar

Resumen. Los analistas usualmente describen requisitos usando notaciones involucrando conceptos técnicos que los clientes suelen desconocer. El expresar requisitos con notaciones legibles para los clientes (NLC) – i. e. que usan conceptos conocidos por ellos – les permite comprenderlos, validarlos, y colaborar en su mantenimiento. La minería de elementos respetando conceptos técnicos (necesitados por los desarrolladores) a partir de modelos NLC es una tarea importante. Actualmente existe falta de NLC apropiados para describir en detalle cómo las cualidades no funcionales (CNF) afectan globalmente al sistema; solo hay algunas notaciones poco detalladas para ello, que solo permiten generar modelos de requisitos orientados a aspectos (OA) muy poco detallados. Por eso proponemos para el dominio de sistemas de información web una NLC que permite expresar cómo CNF afectan globalmente a la parte funcional del sistema. Además definimos una transformación de modelos usando ATL para mapear esos modelos NCU a dos notaciones conocidas de requisitos OA.

Abstract. Analysts usually describe requirements by using modeling notations which involve technical concepts that the clients do not know. Expressing requirements using notations readable by the clients (NLC) – i.e. using concepts known by the clients – allows the clients understanding them, validating them, and collaborating in their maintenance. The mining of elements according to technical concepts (needed by the developers) from NLC models is an important task. At present, there is a lack of appropriate NFCs to describe in depth how non-functional qualities (CNF) affect the system globally; the few available lack sufficient expressiveness for that purpose and only permit the generation of aspect-oriented (OA) requirements with little detail. We propose, for the domain of web information systems, a NLC that allows us to express how CNFs affect globally the functional part of the system. Additionally, we define a model transformation using ATL to map NLC models onto two well-known OA requirement notations.

Palabras Clave: Ingeniería de requisitos, requisitos no funcionales, orientación a aspectos, transformación de modelos, aplicaciones web.

1 Introducción

Para el tratamiento requisitos no funcionales (RNF) durante la ingeniería de requisitos los clientes pueden ayudar a recolectarlos, validarlos y mantenerlos; o los desarrolladores pueden usar su experiencia para definirlos usando notaciones de modelado de requisitos. Cuando los analistas definen los RNF suelen usar notaciones de requisitos basadas en metas (NFR [7], i* [26], KAOS [25], etc.) y/o notaciones de requisitos orientadas a aspectos (OA) - p.ej. AOSDUC [12], método de Sousa etal. [21], AORE con ARCaDE [20], [17, 16], etc. - Cuando los usuarios proporcionan requisitos suelen hacerlo en lenguaje natural o en alguna notación globalmente difundida (p.ej. casos de uso (CU) -[9]-, documentos XML ([10]), etc.) y a partir de los mismos se pueden identificar o generar los elementos de acuerdo con los conceptos de modelado basado en metas o modelado OA. Pero describir requisitos usando notaciones basadas en metas u OA implica basarse en conceptos técnicos (p.ej. softgoal, tópico, contribución, correlación, pointcut, advice, crosscutting, aspecto, etc.) y usar estrategias, técnicas, etc. para su construcción; cosa que los clientes no saben hacer. Esos conceptos técnicos son útiles porque sirven para guiar las fases posteriores de desarrollo (diseño, implementación, etc.).

Las *notaciones legibles por los clientes* (NLC) son notaciones que usan conceptos conocidos por los clientes ya sea porque usan su vocabulario o porque pertenecen a notaciones ampliamente difundidas (p.ej. modelos de CU, descripciones de escenarios, etc.). En ingeniería del software son muy importantes las NLC porque los requisitos expresados a través de ellas pueden ser fácilmente comprendidos (sin necesidad de un entrenamiento), validados, y modificados por los clientes. Además, el mapeo de los requisitos de los clientes en lenguaje natural a estas NLC es más directo que el mapeo a notaciones que involucran conceptos técnicos. La minería de elementos respetando conceptos técnicos a partir de modelos NLC es necesaria, porque dicho tipo de elementos son usados para dirigir el desarrollo de software en etapas posteriores a la ingeniería de requisitos (diseño, implementación, etc.). Dicha minería puede llevarse a cabo manualmente o por medio de herramientas; luego los elementos obtenidos en la minería son aprovechados por los desarrolladores para crear los modelos técnicos en los cuales se van a apoyar (modelos de requisitos basados en metas, OA, etc.)

Los usuarios suelen referirse a cualidades no funcionales (CNF) en sus requisitos en lenguaje natural ya sea expresando escenarios donde aparecen, o refiriéndose a ellas en forma global indicando cómo una CNF impacta sobre funcionalidades y otras CNF del sistema. El primer caso es muy conocido y ha sido estudiado bastante; para el mismo hay notaciones ampliamente difundidas (descripciones de CU, BPMN, etc.); además se ha estudiado cómo a partir de estas notaciones identificar conceptos como aspectos, pointcuts, relaciones de crosscutting, advices, etc. (p.ej. [8]). El tener que dar todos los escenarios involucrando RNF es costoso en tiempo y a este costo se le suma el de procesar dichos escenarios en forma semiautomática para obtener modelos OA o basados en metas. Por eso es interesante el no tener que incurrir en tantos costos y cobra interés el segundo caso donde los usuarios pueden definir en una sola oración cómo una CNF afecta globalmente al sistema; o sea, en una oración se puede plasmar lo que de otra manera involucraría muchos escenarios que muestran vínculos de CNF con requisitos funcionales.

Consideramos que existe la necesidad de NLC apropiados para describir en detalle cómo cada CNF afecta globalmente al sistema; lo que si hay es algunas notaciones poco detalladas para ello que no permiten generar requisitos OA en forma detallada (p.ej. [24, 23, 17]).

Los objetivos de este trabajo son dos: la definición de una NLC para describir cómo las CNF afectan globalmente al sistema y aprovechar esta notación para producir otros modelos de requisitos basados en conceptos técnicos (p.ej.: generar automáticamente modelos OA, ayudar a definir modelos de requisitos funcionales y modelos de RNF).

En un primer estudio documentado en este paper proponemos una NLC para el dominio de sistemas de información web que permite describir el impacto global de CNF en la parte funcional del sistema; se produjo para ello un metamodelo llamado MICNF (Sec. 2). Luego mediante el uso de transformación de modelos en ATL ([13]) planteamos mapear modelos MICNF a modelos de requisitos OA; proporcionamos para ello dos ejemplos: generación de modelos de CU unificados y generación de tablas de composición; ambas son notaciones descritas en [21] y son de utilidad para los desarrolladores en etapas posteriores a requisitos (Sec. 5); por ejemplo esas notaciones fueron usadas en el método AOWE de desarrollo OA de aplicaciones web ([4,5]) que contempla ingeniería de requisitos, análisis y diseño. Además planteamos un proceso de ingeniería de requisitos que integra la creación y generación de todos los modelos usados en este estudio inicial (Sec. 4). En este paper consideramos como caso de estudio un sistema de comercio electrónico.

2 Vinculando CNFs con la Parte Funcional del Sistema

Para definir el metamodelo MICNF tuvimos en cuenta numerosos ejemplos de requisitos de impacto de CNF en el sistema (RICNF) expresados en lenguaje natural (muchos son nuestros y algunos están motivados por ejemplos en [7,23]). De los ejemplos encontrados, deducimos que es muy común que los usuarios cuando expresan un RICNF se refieren a algún tipo de recurso (p.ej. humano, de datos almacenados, de contenido en la interfaz, de servicio externo a la aplicación, funcionalidad, etc.) que aparece en la parte funcional del sistema afectada por el RICNF. A través de este recurso los usuarios hacen referencia implícita a la parcela de todas las unidades de requisitos funcionales (URF) del sistema que hacen referencia a este recurso (una URF puede ser: un CU, un escenario, proceso de negocios, una tarea, etc.). Entonces expresamos la parte del sistema impactada en un RICNF mediante una *parcela del sistema* (PS) definida implícitamente a través de un recurso. Además los RICNF en lenguaje natural suelen expresar de qué modo una CNF afecta a la ejecución de las URF de su PS (i.e. se expresa cuándo tiene lugar el tratamiento de la CNF durante la ejecución de esas URF) y esto se refleja en nuestro metamodelo mediante lo que llamamos *parámetros de impacto*. De acuerdo con estas observaciones definimos el metamodelo MOF MICNF, que puede verse en la Fig. 1. Los elementos de modelado del metamodelo MICNF son:

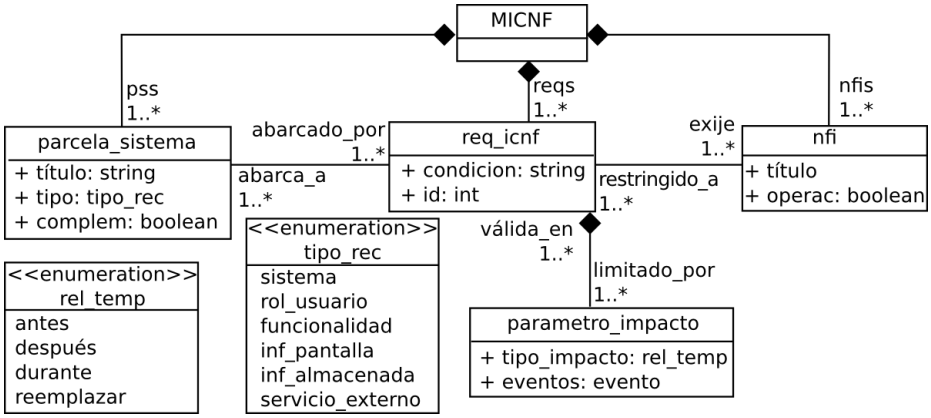

Fig. 1. Metamodelo de Impacto de Cualidades no Funcionales (MICNF)

MICNF: Es una metaclase contenedora para los elementos de modelado.
Parcela del Sistema (PS). Una PS es una parte de nuestro sistema sobre la cual podemos imponer RNF; a una PS se la piensa como un conjunto URF. Las PS suelen ser descritas mediante referencias a recursos de algún tipo. Consideramos los siguientes tipos de recurso:

- **Sistema:** la PS consiste de todos las URF del sistema. Se grafica con un círculo con una S dentro.
- **Funcionalidad:** la PS consiste la URF que tiene por nombre el de la funcionalidad o de todas las URF que contienen en su descripción el nombre de la funcionalidad. Se grafica con un óvalo.
- **Rol de usuario:** la PS consiste de todas las URF en los cuales un rol de usuario interactúa con la aplicación. Se grafica con el símbolo de actor de los modelos de CU.
- **Información en pantalla:** la PS consiste de todas las URF en los que se hace algo con respecto a la información en pantalla (p.ej. mostrarla, modificarla, etc.). Se lo grafica con una pantalla.
- **Información almacenada:** la PS consiste de todas las URF donde se hace alguna operación con la información almacenada (p.ej. agregar, borrar, etc.). Se lo grafica con una hoja escrita.
- **Servicio externo:** la PS consiste de todas las URF que requieren la ejecución de un servicio externo (p.ej. servicio de un sistema externo, servicio web, etc.). Se grafica con un par de engranajes.

Cuando para el tipo información en pantalla se usa el título ANY la PS significa: todas las URF en las que se hace algo con respecto a alguna información en pantalla. Cuando para el tipo información almacenada se usa el título ANY la PS significa todas las URF en las que se hace alguna operación de actualización de alguna información almacenada.

Se puede representar la PS que es el complemento de una PS *p* (i.e. todas las URF del sistema menos las de la PS *p*) dibujando la PS *p* precedida por el símbolo ¬ . Para reflejar si una PS es de tipo complemento, se agregó en la metaclase *parcela-sistema*

el metaatributo *complemento*. Por ejemplo, para un sitio de correo electrónico tenemos el MICNF (ver Fig. 2 (h)): para cualquier operación que no sea finalizar (PS) después que se presiona help se *muestra información de ayuda* (NFI).

NFI: La clase NFI (ítem no funcional) representará tanto CNF (las cuales se grafican con nubes de contorno fino como en el framework NFR) como soluciones concretas que permiten alcanzar CNF (las cuales se grafican con nubes de contorno grueso), p.ej. operaciones, decisiones de diseño, decisiones de arquitectura, etc.

Parámetro de Impacto: Este elemento sirve para indicar en un RICNF cuándo y de qué manera precisa un NFI va a impactar en la PS. Un parámetro de impacto se grafica mediante el símbolo de un reloj analógico y al lado del mismo se ponen los valores de sus atributos. Para un parámetro de impacto se consideran dos atributos: *evento* para indicar cuándo ocurre el impacto y que representa uno o más eventos en la URF, y *tipo de impacto* que indica cómo impacta un NFI en la PS y puede ser: *antes*, *después*, *durante* y *reemplazar* (estos se parecen a los operadores de composición del área de OA). Los casos posibles son:

- **antes|después evento:** el NFI se cumple justo antes/después del evento**.**
- **reemplazar evento**: el NFI se cumple justo luego del evento y a partir de ahí reemplaza a la ejecución URF que estaba en ejecución.
- **durante evento 1, evento 2**: el NFI se cumple durante cada intervalo dado por los eventos 1 y 2 (en ese orden) de la ejecución de la URF (de la PS).

req_ICNF: Este elemento de modelado se va a usar para describir un RICNF, el cual explica de qué modo uno o más NFI afectan a una PS. Notar que un RICNF puede referirse a varios recursos y a varios parámetros de impacto a la vez, con el significado de que las NFI impactan sobre la PS dada por la intersección de las PS asociadas a los recursos y la forma de impacto viene dada por la suma de los parámetros de impacto. Un req_ICNF tiene un atributo *condición* que representa una condición que debe cumplirse en el sistema para poder ser impactado por el elemento NFI; esa condición debe estar encerrada entre corchetes. Un req-ICNF se lo grafica con una flecha con varias puntas triangulares, una por cada PS y con varias colas, una por cada NFI; arriba de la flecha se suelen poner los parámetros de impacto y abajo la condición.

Los eventos de un parámetro de impacto pueden ser eventos concretos o eventos genéricos. Un *evento genérico* representa un conjunto de eventos del mismo tipo. Los eventos genéricos considerados son: *iniciar* (primer evento de las URF), *terminar* (último evento de las URF), *input* (un usuario produce un input), *output* (el sistema produce un output), *alterar* <dato> (el sistema inserta/borra/ modifica el <dato>), *modificar* | *insertar* | *borrar* (el sistema modifica/inserta/borra algún dato), *envío* (el sistema envía datos a una aplicación externa), *recepción* (el sistema recibe respuesta de alguna aplicación externa).

Ahora damos algunos ejemplos en lenguaje natural de RICNF para un sistema de comercio electrónico que se modelan en la Fig. 2. Para mostrar lo directa que es la traducción al modelado se subrayaron las PS y se pusieron en itálico las NFI.

(a) Mientras se realiza una compra o se hace una evaluación de producto se muestran *ayudas sobre la operación del sitio*.
(b) Después de una operación que modifica alguna información se *guarda un registro log* para la operación realizada.
(c) Luego que se muestra la información de un producto al cliente, se *muestran productos relacionados con ese producto*.

(d) Mientras se hace alguna operación que muestra el catálogo en pantalla se dan *ayudas a la orientación del usuario*.
(e) Si durante una compra el usuario está llenando los datos de compra o los datos de envío y presiona cancelar entonces se *cancela* la compra. Si durante el llenado de datos de evaluación de un producto el usuario presiona cancelar, entonces se *cancela* la creación de una evaluación de producto.
(f) Antes de iniciar sesión en el caso que el sitio esté sobrecargado o inaccesible se elige *usar un sitio espejo*.
(g) Hay que *validar login y password* antes de comenzar compra, antes de cerrar una cuenta de usuario y antes de modificar informaciones del catálogo.

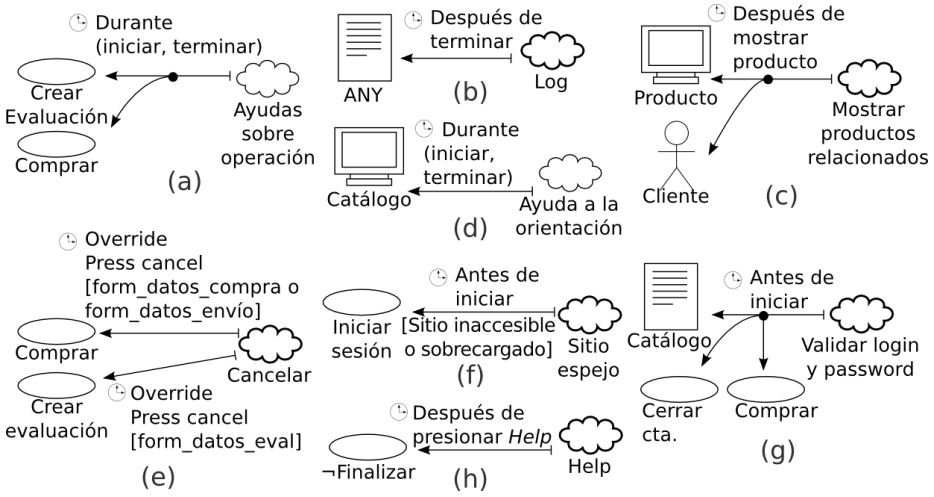

Fig. 2. Ejemplos de RICNF

3 Otros Modelos de Requisitos Contemplados

En esta sección decidimos y definimos los metamodelos necesarios para el desarrollo de este trabajo. En este paper trabajamos con URFs que son CU, por lo tanto nos apoyamos en el metamodelo de UML 2.0 para los diagramas de CU. Dado que será preciso manipular estas descripciones de CU mediante transformaciones, es necesario describirlas en algún espacio tecnológico. Por los motivos explicados en la sección 5 decidimos usar una notación textual para describir CU, los cuales se describen de acuerdo con el DTD[1] siguiente:

```
<!ELEMENT CUs (descripcionCU+)>
<!ELEMENT descripcionCU (base,descripcion,alternativa*)>
<!ELEMENT descripcion (pasodesc+)>
```

1 Por limitaciones de espacio, se han omitido los elementos de tipo #PCDATA

<!ELEMENT pasodesc (actor_desc, accion_desc, objeto_desc,
destinatario_desc*, condicion_desc*)>
<!ATTLIST pasodesc numpasodesc CDATA #REQUIRED>
<!ATTLIST accion_desc io_desc (entrada|salida|otro) #IMPLIED>
<!ELEMENT base (nombre, resumen, actor+, pre*, post*)>
<!ATTLIST base id CDATA #REQUIRED>
<!ELEMENT alternativa (num_alt, condicion+, pasodesc+)>
<!ATTLIST alternativa viene_de CDATA #REQUIRED>
<!ATTLIST alternativa sale_en CDATA #REQUIRED>

Para los SIG del framework NFR [7], produjimos un metamodelo llamado MNFR (ver Fig. 3), el cual es una simplificación del metamodelo en [23].

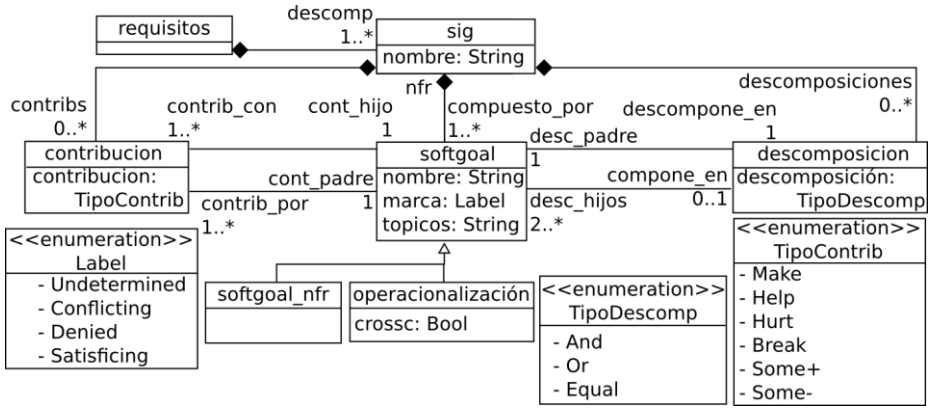

Fig. 3. Metamodelo MNFR

Para el modelado de requisitos OA usamos tanto el Modelo de CU Unificado como las tablas de composición del método OA de Sousa etal. ([21]). Como no encontramos metamodelos para las tablas de composición, definimos en este paper un metamodelo MOF para las mismas (Ver Fig. 4).

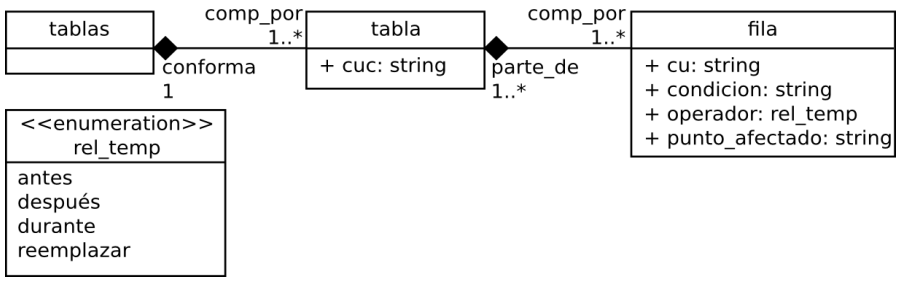

Fig. 4. Metamodelo de Tablas de composición.

4 Proceso de Ingeniería de Requisitos

En la Fig. 4 se muestra un esquema del proceso de ingeniería de requisitos adoptado en este paper. Primero a partir de diversas fuentes (clientes, usuarios finales, documentos de proyectos similares, etc.) se obtienen RICNF descritos en lenguaje natural, luego estos requisitos son traducidos a un modelo MICNF. A partir del modelo MICNF el analista identifica softgoals y se apoya en ellos para definir el modelo NFR; para esto se usa la idea de que a partir de un RICNF r que tiene PS $p_1, p_2, ..., p_n$ y un NFI n, se puede generar el softgoal: $n[p_1 \cap p_2 \cap ... \cap p_n]$ donde el tópico es la intersección de los conjuntos de URFs de las PS p_i. Un softgoal $n[t]$ obtenido de RICNF puede refinarse a nuevos softgoals ya sea en tipo (considerando aspectos más específicos o detallados de n) o en tópico (considerando subconjuntos de t). Luego a partir del modelo MICNF el analista saca algunos CU (a partir de las PS funcionalidad) y actores (a partir de PS rol de usuario y servicio externo) que ayudarán para definir el modelo de CU y con la ayuda de clientes y usuarios completa dicho modelo. A partir del modelo MICNF el analista obtiene algunas informaciones que servirán para elaborar descripciones de CU: informaciones de PS como funcionalidades, informaciones en pantalla, informaciones almacenadas, etc. y eventos de parámetros de impacto; luego con la ayuda de clientes y usuarios se elaboran las descripciones de los CU (estas descripciones no contienen referencias a RNF, pues dicha información está en el modelo ICNF). Finalmente a los modelos MICNF, NFR, de CU y de descripciones de CU se les aplica una transformación llamada Req2ReqOA que genera el modelo de CU unificado y las tablas de composición. Observar que los modelos MICNF, de CU y de descripciones de CU pueden ser evaluados por el cliente; esto es posible por la amplia difusión de los modelos de CU, de las descripciones textuales de CU, de XML y por ser el modelo ICNF una NLC.

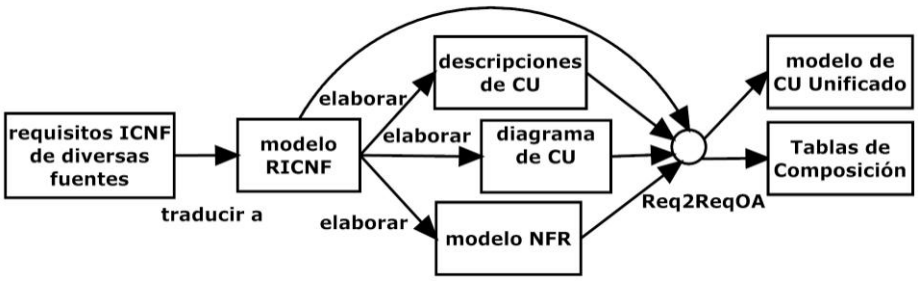

Fig. 5. Proceso de Ingeniería de Requisitos OA

En la Fig. 6 se muestra un SIG para usabilidad en el que se tienen en cuenta los RICNF descritos en la Fig. 2. Notar que si un tópico corresponde a un tipo de PS que no es funcionalidad, entonces al nombre del tópico le anteponemos el tipo de PS, expresado mediante una abreviatura de dos caracteres (a esta información la necesitamos durante la aplicación de la transformación Req2ReqOA). A partir de los RICNF de la Fig. 2 se identificaron los CU: iniciar sesión, comprar, crear evaluación, cerrar

cuenta y el actor cliente. El modelo de CU contiene otros CU como: inspeccionar catálogo, buscar producto, crear cuenta, mirar carrito, ver status de pedido, ordenar envío, procesar pago, etc. y otros actores como: empleado, sistema mercantil, etc.

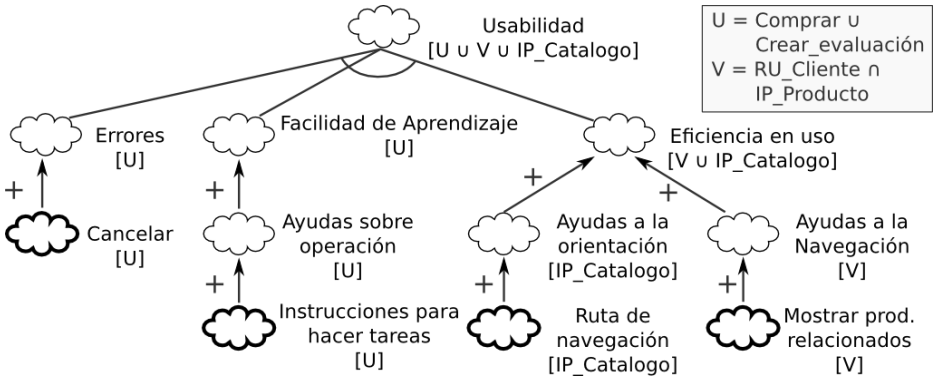

Fig. 6. SIG de Usabilidad para sistema de comercio electrónico

5 Generación de Modelos de Requisitos OA

Ahora estudiamos la transformación Req2ReqOA (Fig.5) desde modelos MICNF, de CU, de descripciones de CU y MNFR a los modelos (de requisitos OA) de CU unificado y las tablas de composición (de Sousa etal. [21]). Durante esta transformación solo se tienen en cuenta las operaciones necesarias para cumplir con una CNF y no otras facilidades para tratarla como decisiones de arquitectura, de diseño, etc. La razón de esto es que los modelos de CU unificado y tablas de composición solo contemplan operaciones que ayudan a cumplir con CNFs. Un diagrama que muestra cómo esta transformación se divide en reglas de transformación puede apreciarse en la Fig. 7. Las reglas usadas durante el proceso de transformación se describen en la tabla 1.

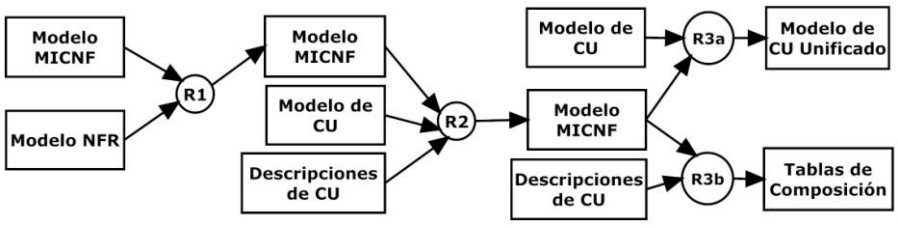

Fig. 7. Diagrama de transformaciones

Tabla 1. Reglas de Transformación

R1	Condiciones de Aplicabilidad	Todo NFI del modelo MICNF se corresponde con un softgoal del modelo MNFR.
	Descripción	Cada RICNF *r* que involucra un NFI *n* es reemplazado por RICNF $r_1,\ldots,r_m$ involucrando el parámetro de impacto de *r* y operaciones NFI $n_1,\ldots,n_m$ correspondientes a un conjunto de operacionalizaciones en el MNFR que sirven para satisfacer el softgoal asociado a *n*.
R2	Descripción	Todo RICNF *r* del MICNF de entrada se reemplaza por un RICNF *r'* de modo que *r* es igual a *r'* salvo en las PS. Las PS de *r'* se obtienen así: primero se intersecan las PS de *r* (pensadas como conjuntos de CU del modelo de CU) y luego de la intersección se descartan aquellos CU que no contienen los eventos de ninguno de los parámetros de impacto de *r*.
R3a	Condiciones de Aplicabilidad	En el modelo MICNF de entrada solo aparecen PS funcionalidad que son CU del modelo de CU y solo aparecen NFI operaciones.
	Descripción	Por cada NFI operación en el MICNF de entrada se crea un CU y ese CU se vincula con los CU que impacta (en el MICNF de entrada) ya sea usando una relación <<crosscut>> o una relación <<extend>> de acuerdo con las reglas de Souza para determinar el tipo de relación. Además el modelo de CU unificado contiene al modelo de CU.
R3b	Condiciones de Aplicabilidad	En el modelo MICNF de entrada solo aparecen PS funcionalidad que son CU del modelo de CU y solo aparecen NFI operaciones.
	Descripción	Sea NFI operación *n* del MICNF de entrada *M,* entonces se construye una tabla *T*. Para cada RICNF *r* en M que contiene a *n* se crean filas en *T* como sigue*:* para cada parámetro de impacto *p*, PS funcionalidad CU *c,* y lugares *ls* en descripción de c donde ocurren los eventos de p – un *lugar* consiste de número de paso en un escenario- se crea una fila de T que contiene: el CU c, la condición de r, tipo de impacto de p y *ls*.

Herramientas y Lenguajes de Transformación Empleados. ATL ([13]) es un lenguaje de transformación adecuado para este trabajo, porque permite transformar modelos que respetan metamodelos en MOF, porque es posible referirse en forma declarativa a los elementos de los modelos de entrada (por ejemplo, para la regla 1 para decir que queremos todas las operacionalizaciones operación descendientes de un softgoal que sirven para satisfacerlo conviene ser declarativos para tener una implementación cercana a la especificación de la regla y por ende fácilmente comprensible), y porque debido a que ATL usa OCL se puede usar navegación para referirse a colecciones de elementos en forma compacta, cosa que no sería posible en un enfoque solo declarativo (esta facilidad sirve para copiar colecciones de elementos del modelo de entrada al modelo de salida, y esto es lo que hicimos frecuentemente al codificar la transformación Req2ReqOA).

Queremos que las descripciones de los CU las pueda leer cualquier cliente. Las notaciones de UML usadas en la literatura para describir CU (p.ej. diagramas de secuencia, statecharts, de colaboración, etc.) suelen no ser conocidas por los clientes. Además queremos poder consultar un documento con todas las descripciones de CU para obtener todos aquellos CU cuya descripción cumple con alguna propiedad del

tipo: existe un paso x tal que P(x) y esto se logra si las descripciones de CU se hacen respetando un esquema de algún lenguaje de consultas. Usar descripciones de CU en XML permite que cualquiera pueda leerlas (debido a la simplicidad de XML y de su uso generalizado) y también hacer consultas (usando XPath y XQuery – ver [3]) Por otro lado definir un metamodelo MOF y una sintaxis concreta para descripciones de CU produciría un resultado que no es ampliamente conocido. Entonces las consultas de descripciones de CU en XML las hacemos utilizando XQuery.

Regla 1. Describimos esta regla en bastante más detalle que en la Tabla 1. Sea un RICNF *r* que involucra un NFI *n* y *n* es mapeado en el modelo MNFR a un softgoal $n[t]$. Sean en el modelo MNFR $n_1[t_1],\ldots, n_m[t_m]$ un conjunto de operacionalizaciones que permiten satisfacer $n[t]$. Entonces se reemplaza *r* de MICNF por nuevos RICNF como sigue: para cada t_i $(1 \leq i \leq m)$ se generan nuevos RICNF de la siguiente manera: asumimos que t_i es de la forma $p_1 \cup \ldots \cup p_s$ donde los p_j son de la forma $q_{j,1} \cap \ldots \cap q_{j,nj}$ tal que los $q_{j,k}$ $(1 \leq k \leq n_j)$ son PS con los tipos de recurso encontrados en MICNF, entonces se agregan al modelo MICNF los requisitos r_j (con $1 \leq j \leq s$) compuestos por: el NFI n_i, los parámetros de impacto de *r* y las PS $q_{j,1},\ldots, q_{j,nj}$.

Regla 2. De todas las reglas de la tabla 1 la más compleja es la regla 2, la cual se dividió en dos reglas menores 2.1 y 2.2 (ver Fig. 8) que explicamos.

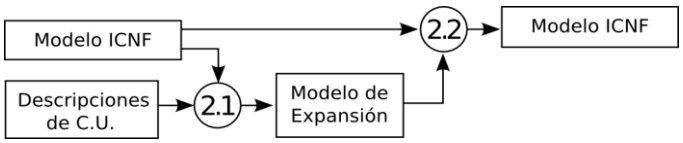

Fig. 8. Regla 2.

Regla 2.1. Se recopila la información de qué CU pertenecen a cada PS, usándose para ello XQuery. El resultado de la regla 2.1 es un modelo intermedio llamado *modelo de expansión* (MEx). Los modelos MEx fueron definidos en MOF y su sintaxis acorde con XMI se expresa en DTD mediante:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Requisitos (parcela_sistema+)>
<!ELEMENT parcela_sistema (titulo, tipo, CU+)>
<!ATTLIST tipo (sistema | rol_usuario | funcionalidad | inf_pantalla | inf_almacenada
| servicio externo)>
<!ELEMENT CU (titulo)> <!ELEMENT titulo (#PCDATA)>
```

La creación del modelo MEx usa consultas XQuery sobre el documento de descripciones de CU para las PS de tipo: información en pantalla, información almacenada, funcionalidad, servicio externo y funcionalidad. Además la creación del modelo MEX usa consultas ATL sobre el modelo de CU para las PS de tipo: sistema, actor y funcionalidad.

Regla 2.2. Primero se usa ATL para transformar cada RICNF *r* del MICNF de entrada en un RICNF *r'* del siguiente modo: se calcula la intersección de los conjuntos de CU asociados por el MEx a las PS de *r,* la cual produce un conjunto de CU *C;* luego *r'* es igual a *r* en todo salvo en que su conjunto de PS consiste de una PS funcionalidad por cada CU de C. El resultado de la transformación anterior es un MICNF M'. Segundo, en cada RICNF *r'* de *M'* se descartan aquellas PS funcionalidad cuyo CU no contiene en su descripción los eventos de ninguno de los parámetros de impacto de *r'*. Para averiguar si los eventos de algún parámetro de impacto de *r'* están en la descripción de un CU se usa XQuery en el texto de la descripción del CU.

6 Trabajo Relacionado

Un tipo interesante de trabajos son aquellos que a partir de un modelo de requisitos no OA permiten identificar ya sea automática o semiautomáticamente aspectos (candidatos) y reflejar su comportamiento en modelos de requisitos OA.

Existen varios enfoques que parten de modelos i* ([26]) para identificar aspectos candidatos [22, 19, 2, 18]. El problema de partir de i* es que es un lenguaje que involucra numerosos conceptos técnicos, no captura los parámetros de impacto ni las PS de los RICNF y a partir de ellos solo se pueden obtener modelos de requisitos OA más pobres que los considerados en este paper.

En [8] a partir de descripciones de CU, diccionarios de sinónimos, listas de palabras clave de RNF se identifican aspectos candidatos y se establecen relaciones crosscutting entre los aspectos candidatos y los CU donde fueron identificados. Nosotros en lugar de partir de descripciones de CU conteniendo referencias a RNF partimos de modelos ICNF y descripciones de CU sin referencias a RNF. Además producimos tablas de composición que son más detalladas.

En [24] se parte de matrices de dependencias entre unidades de software (que pueden ser URF) e intereses (que pueden ser CNF) y a partir de estas matrices se identifican aspectos candidatos y se obtiene la información de qué aspectos candidatos afectan a qué unidades de software. Nuevamente esta notación de matrices es menos expresiva que la de los RICNF al no contemplarse parámetros de impacto ni PS.

Otro tipo de aportes son los modelos de requisitos que permiten describir las relaciones entre RNF y requisitos funcionales. Existen varios enfoques de modelado basados en metas entre ellos: KAOS [25], NFR [7], i* [26], GRL [11], [14, 1, 15], etc. Pero ellos no consideran facilidades como los parámetros de impacto RICNF, las PS de los RICNF y además involucran numerosos conceptos técnicos que los clientes suelen no conocer.

En [23] se propone un lenguaje de modelado integrado extendiendo UML para poder relacionar elementos de modelos de CU con softgoals del framework NFR; esto hace que se puedan asociar softgoals a: sistema (softgoals globales), a CU (softgoals específicos a CU), a actores (softgoals que restringen a actores), a relaciones entre actores y CU (softgoals que restringen la interfaz entre actores y el sistema). Entonces se tienen algunas PS, aunque son menos y menos detalladas que las de los RICNF (p.ej. no está información almacenada e información en pantalla); además en este enfoque no se consideran parámetros de impacto como en RICNF.

Existen numerosas notaciones de modelado de requisitos OA (p.ej. [12, 21, 17, 20, 16]), pero estas suelen incluir conceptos técnicos de OA que los clientes usualmente desconocen; además, no contemplan PS definidas a través de recursos como en los RICNF.

7 Conclusión y Trabajo Futuro

Para definir el metamodelo MICNF consideramos numerosos ejemplos de RICNF en lenguaje natural y definimos dicho metamodelo de modo que el mapeo de esos requisitos en lenguaje natural a modelos en MICNF sea directo.

El enfoque empleado sirve para sistemas de información web 1.0, sobre todo los intensivos en datos. Esto se debe principalmente al tipo de PS que contemplamos y por la forma en que se describen los CU en el DTD.

Para probar nuestro enfoque consideramos dos casos de estudio: uno de comercio electrónico y otro de correo electrónico; en este paper por limitaciones en el espacio mostramos solo una pequeña parte del ejemplo de comercio electrónico. De acuerdo a los casos de estudio realizados observamos que el modelo MICNF puede ser bastante bueno como base para producir los SIG (en nuestros ejemplos la mayoría de las ramas de los SIG suelen contener más bien hacia el final – i.e en hojas o padres de hojas- un softgoal asociado a un NFI del modelo MICNF). Además, los RICNF para el ejemplo de comercio electrónico contienen el 50 % de los elementos del modelo de CU y para el sistema de correo electrónico contienen el 16% de los elementos del mo-delo de CU; esta variación se debe a que de un ejemplo a otro puede variar mucho la cantidad de PS del tipo funcionalidad y rol de usuario que son usadas en los RICNF.

Como trabajo futuro se puede extender MICNF para que CNF puedan impactar otras CNF. Planeamos extender MICNF para otro tipo de aplicaciones (p.ej. sistemas de control, Web 2.0, etc.)

Hemos estudiado el mapeo de modelos MICNF a modelos OA sencillos: modelos de CU unificados, tablas de composición. El mapeo de modelos ICNF a modelos OA más complicados, por ejemplo a aquellos que involucran pointcuts, parece ser a simple vista posible, y planeamos chequearlo en el futuro.

Referencias

1. Akoumianakis, D., Katsis, A., Vidakis, N.: Non-functional User Interface Requirements Notation (NfRn) for Modeling the Global Execution Context of Tasks. In: K. Coninx, K. Luyten, and K.A. Schneider (Eds.) TAMODIA 2006, LNCS, vol. 4385, pp. 259--274. Springer Berlin / Heidelberg (2006)
2. Alencar, F., Silva, C., Moreira, A., Araújo, J., Castro, J.: Identifying Candidate Aspects with I-star Approach. In: Workshop on Early Aspects at AOSD 2006, pp. 4--10. Bonn, Germany (2006)
3. Brundage, M.: XQuery: the XML Query Language. Addison Wesley (2004).
4. Casalánguida, H., Durán, J. E.: AOWE: una Metodología Orientada a Aspectos para Desarrollo de Aplicaciones Web. 8th Argentinean Symposium on Software Engineering, pp. 90--104 (2007)

5. Casalánguida, H., Durán, J. E.: Modelado Orientado a Aspectos de Navegación para Aplicaciones Web basado en UML. IEEE Latin America Transactions, Vol. 7, No. 1, 92--100 (2009)
6. Chen, K., Zhao, H., Zhang, W., Mei, H.: Identification of Crosscutting Requirements Based on Feature Dependency Analysis. In: ICRE 2006, pp. 300--303 (2006)
7. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non functional Requirements in Software Engineering. Kluwer Academic Publisher, Boston (2000)
8. Haak, B., Díaz, M., Marcos, C., Prior, J.: Aspects Extractor: Identificación de Aspectos en la Ingeniería de Requerimientos. In: IDEAS 06, 9° Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software (2006)
9. Hamilton, K., Milles, R.: Learning UML 2.0. O'Reilly (2006)
10. Hunter, D., Cagle, K., Dix, C.: Beginning XML: XML Schemas, SOAP, XSLT, DOM and SAX 2.0. Wrox Press (2003)
11. ITU - Telecommunication Standardization Sector Draft Specification of the Goal-oriented Requirement Language (Z.151), September 2001
12. Jacobson, I., Pan-Wei, Ng.: Aspect-Oriented Software Development with Use Cases. Addison-Wesley Professional (2004).
13. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A Model Transformation Tool. Science of Computer Programming vol. 72, Nr 1, 31--39 (2008)
14. Jureta, I., Faulkner, S., Schobbens, P-Y.: A More Expressive Softgoal Conceptualization for Quality Requirements Analysis. In: Embley, DW, Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 281—295, Springer Berlin / Heidelberg (2006)
15. Kassab, M. Ormandjieva, O., Daneva, M.: An Ontology Based Approach to Non-functional Requirements Conceptualization. In: 4th Intl. Conf. on Software Engineering Advances, pp. 299--308 (2009)
16. Moreira, A. and Araújo, J., Handling Unanticipated Requirements Change with Aspects. In: Software Engineering and Knowledge Engineering Conf. (SEKE'04), Banff, Canada, (2004)
17. Moreira, A., Araújo, J., Rashid, A.: A Concern-Oriented Requirements Engineering Model. In: Pastor, O., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 293–308. Springer, Heidelberg (2005)
18. Monteiro, C., Ramos, R., Castro, J., Araújo, J., Moreira, A., Alencar, F.: The iAspectPlugin to Automatize the Identification of Crosscutting Concerns on i*. In: 1st Latin-American Workshop on Aspect-Oriented Software Development at SBES 2007, pp. 105--116. Sociedade Brasileira de Computação (2007)
19. Niu N., Easterbrook S.: Discovering Aspects in Requirements with Repertory Grid. In: International Workshop Early Aspects at ICSE 2006, pp. 35--42. ACM Press (2006)
20. Rashid, A., Moreira, A., Araújo, J.: Modularisation and Composition of Aspectual Requirements. In: Intl. Conf. on AOSD, pp. 11--20 (2003)
21. Sousa, G., Soares, S., Borba, P., Castro, J.: Separation of Crosscutting Concerns from Requirements to Design: Adapting a Use Case Driven Approach. In: Early Aspects Workshop at AOSD 2004, pp. 5—15 (2004)
22. Spies, E., Rüger, J. and Moreira, A.: Using i* to Identify Candidate Aspects. In: Workshop on Aspect-Oriented Modeling at UML'04, Lisbon, Portugal (2004)
23. Supakkul, S., Chung, L.: A UML Profile for Goal-Oriented and Use Case-Driven Representation of NFRs and Frs. In: 3rd ACIS Intl. Conf. on Software Engineering, Research, Management and Applications, pp. 112—121, IEEE Computer Society (2005)
24. van den Berg, K., Conejero, J. M., Hernández, J.: Identification of Crosscutting in Software Design. In: 8th Intl. Workshop on Aspect-Oriented Modeling, Bonn, Germany (2006)
25. van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: ICRE 2001, pp. 249--261. IEEE Computer Society, Washington DC (2001)
26. Yu, E.: Towards Modeling and Reasoning Support for Early Requirements Engineering. In: ICRE 1997, pp. 226–-235. IEEE Computer Society, Washington DC (1997)