Generación Automática de Casos de Uso para Desarrollo de Software Basado en Reutilización*

Oscar López[1]**, Miguel Ángel Laguna[2] y José Manuel Marqués[2]

[1]Instituto Tecnológico de Costa Rica
olopez@infor.uva.es
[2]Universidad de Valladolid
{mlaguna, jmmc} @infor.uva.es

Resumen. El desarrollo de software con reutilización se basa en la selección adecuada de elementos reutilizables del repositorio. La generación automática de casos de uso es una opción para acelerar la definición precisa de los requisitos funcionales como paso inicial para el desarrollo de software con reutilización. Por lo anterior, en este artículo proponemos un marco de trabajo para normalizar la captura de requisitos en forma de casos de uso que puedan expresarse automáticamente en forma de assets de análisis viables para formar mecanos en el contexto de la reutilización sistemática. La normalización de los requisitos se realiza mediante un proceso que involucra el uso de workflows, grafos de casos y redes de Petri para tratar analíticamente la funcionalidad del sistema. Los grafos de casos que proponemos son una aproximación para modelar la dinámica del negocio y lograr un incremento en el potencial de la reutilización.

Palabras Clave. Ingeniería de requisitos, casos de uso, reutilización del software, workflow, redes de Petri.

1 Introducción

El desarrollo de software con reutilización pretende la mejora sustancial en la productividad y calidad de sistemas software [15]. El proceso con reutilización se inicia con la selección de los elementos reutilizables idóneos almacenados en un repositorio [9]. De esa manera, la reutilización sistemática del software enfatiza la necesidad de la definición pronta y precisa de los requisitos. En el presente trabajo describimos un marco para la generación temprana y automatizada de los requisitos del software para aprovechar las ventajas de la reutilización.

La ingeniería de requisitos es un proceso especializado que se realiza en un dominio para documentar las características que debe cumplir un producto software y transformarlas en una especificación [14][10]. El documento de especificación de requisitos debe cubrir la representación y comprensión del ambiente específico y debe

* Parcialmente financiado por CICYT(TIC97-0593-c05-05) como parte del proyecto MENHIR.
** Oscar López es patrocinado por la AECI en España y el ITCR y el MICYT en Costa Rica

contener todas las funciones esenciales (funcionalidad delimitada) del software [15] de manera rastreable, no ambigua, con independencia de los requisitos no funcionales y de las restricciones de diseño [6]. La reutilización del software exige que la especificación permita comparar y adaptar requisitos, Prieto-Díaz [16].

La expresión de requisitos mediante el lenguaje natural no satisface las necesidades de especificación para desarrollo con reutilización. Los problemas de ambigüedad, escalabilidad y trazabilidad son inherentes al lenguaje natural [12]. Además, la documentación de los requisitos de usuario en lenguaje natural presenta una escasa conexión con la reingeniería del negocio y con la implementación del sistema [2].

Los casos de uso (UC) y los casos de uso del negocio (BUC) [7][8] son una estrategia de captura de requisitos ampliamente aceptada frente a las deficiencias del lenguaje natural. Los UC resuelven las deficiencias de escalabilidad y trazabilidad, y proporcionan facilidad para la descripción. Sin embargo, de acuerdo con Lee, Cha y Kwon [12], los UC no superan la ambigüedad del lenguaje natural. Nosotros además sostenemos que los casos de uso no proveen medios para la comparación y adaptación de requisitos en desarrollo con reutilización. Cockburn [2], agrega que aunque la idea intuitiva de la *forma en la cual el usuario utiliza el sistema* es muy aceptada, no siempre se concreta de igual manera por todos los interesados en el desarrollo. El resultado es la proliferación de definiciones de casos de uso y la disparidad de las aproximaciones para especificación de requisitos con base en casos de uso.

Por todo lo anterior, en el grupo GIRO[1] nos hemos planteado la necesidad de normalizar la captura de requisitos mediante casos de uso. En trabajos previos hemos propuesto determinar la funcionalidad inicial del sistema software a partir de la descripción del quehacer del usuario [11] y derivar los casos de usos en concordancia con algún formalismo sintáctico y semántico [13]. El presente trabajo muestra un mecanismo formal y automático para generar los requisitos funcionales como una colección de escenarios de interacción entre el usuario y el sistema.

Nuestro objetivo central es contar con una estrategia rápida para determinar requisitos funcionales de un sistema de información y almacenarlos en forma de elementos reutilizables (assets) de nivel de análisis. El punto de partida es un flujo de trabajo (*workflow*) modelado como un Grafo de Casos que conduce hacia la generación automática de casos de uso que pueden ser incluidos en un repositorio para formar estructuras complejas de reutilización llamadas mecanos [5]. De esta manera las bondades de los casos de uso se refuerzan mediante el uso de workflows lo que permite la escalabilidad y la trazabilidad. También, al corregir la informalidad de los casos de uso mediante el formalismo de las redes de Petri, estaremos en posibilidad de obtener mecanismos para comparar requisitos. El marco establecido es claro y permite modelar el negocio con un mínimo de información.

El resto del documento se organiza así: La sección 2 presenta un marco de referencia para generar casos de uso y assets. La sección 3 especifica el proceso para modelar el problema mediante workflows y así automatizar la captura de requisitos funcionales como casos de uso. La sección 4 concluye el artículo e indica acciones de trabajo futuro.

[1] Grupo de Investigación en Reutilización y Orientación al Objeto, Universidad de Valladolid.

2 Un Marco General para la Generación de Casos de Uso

La definición de la funcionalidad del software se realiza dentro de un complejo esquema de comunicación entre un usuario, a veces inseguro de la funcionalidad requerida, y el analista de sistemas o el ingeniero de requisitos, que deberá especificar correctamente la funcionalidad mediante aproximaciones sucesivas [15]. Esta interacción usuario-analista requiere de una estrategia robusta que garantice que los requisitos serán descubiertos con precisión y que serán expresados de forma correcta y sin ambigüedad, que sea verificable, trazable y modificable [11].

Además de las dificultades para obtener los requisitos correctos a partir del análisis de requisitos, es un hecho aceptado que la obtención de los mayores beneficios de la reutilización pasa por la comprensión precoz de la funcionalidad del sistema. Para eso es conveniente contar con un modelo de proceso para elicitación (descubrimiento) de requisitos como el que se presenta en la figura 1. Este modelo se sitúa en el contexto de los sistemas de eventos discretos, de acuerdo con Silva [19], puesto que el interés se centra en la evolución de los estados sin importar cuándo el sistema alcanza un estado particular, ni cuánto tiempo permanece en tal estado. Es decir, el modelado de los requisitos en esta etapa está centrado en la secuencia de estados del sistema. Para ello se hará uso de las redes de Petri que proporcionan un soporte formal para la verificación del comportamiento lógico del sistema, [20][12].

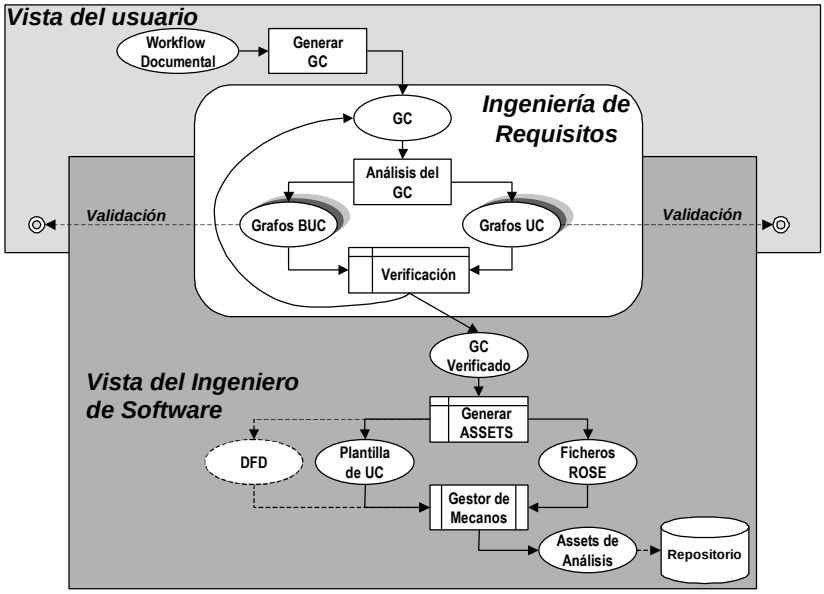


Fig. 1. Marco general de trabajo para la normalización de la captura de los requisitos de usuario.

El modelo de proceso establece dos niveles relacionados para la elicitación de los requisitos: El nivel del usuario y el nivel del Ingeniero de Software. El primero con una visión externa (caja negra) del sistema, y el segundo con la vista interior de esa

caja negra. Dentro del nivel del Ingeniero del Software existe la vista del Ingeniero de Requisitos la cual actúa como una interfaz entre los dos niveles anteriores.

El usuario proporciona la primera aproximación de los requisitos funcionales del sistema expresada como un workflow documental que según los estándares correspondientes de la Workflow Managment Coalition (WfMC) [21] se han utilizado con éxito para representar la logística de los procesos de negocio. Se utiliza un tipo específico y sencillo de un workflow, conocido como diagrama Documentos-Tareas (dDT) [3] que, si es utilizado de forma rigurosa, cumple con los estándares de la WfMC en tanto que especifica qué tareas se deben realizar y en qué orden. Ese workflow inicial permite modelar el flujo de información desde que esta ingresa en el sistema hasta que sale debidamente transformada. En esta etapa la propuesta metodológica proporciona una definición preliminar de los requisitos de usuario.

La labor de Ingeniería de Requisitos en este marco consiste en modelar la funcionalidad del sistema a través de un Grafo de Casos (GC) lo que será detallado en la Sección 3. Del análisis del GC se obtendrán familias de casos de uso del negocio (Grafos BUC) y familias de casos de uso (Grafos UC). Las actividades de validación y verificación permitirían corregir la versión inicial del GC. De ser necesario, el proceso puede solicitar una nueva versión del GC y los Grafos BUC y los Grafos UC.

El Ingeniero del Software puede utilizar el GC verificado para generar los assets que pueden ser expresados en forma de plantillas de casos de uso, como en Durán [4], ficheros ROSE [1][17] o Diagramas de Flujos de Datos (DFD). Estos assets pueden ser recibidos por un gestor de mecanos para interactuar con el repositorio y recuperar los mecanos correspondientes para el desarrollo de software con reutilización.

3 Modelado del Problema

3.1 El Grafo de Casos

Un dDT es un tipo particular de workflow que modela la logística de los procesos del negocio. Con alguna información adicional, el dDT se convierte en un grafo de casos que especifica un flujo de tareas que se deben ejecutar en un orden determinado. Las tareas y los documentos enlazados con arcos son la base de un grafo de casos.

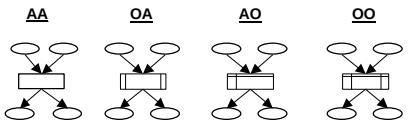
Definición 1. **Grafo de Casos:** Un Grafo de Casos es una cuádrupla (D,T,A,E), donde:

- D es un conjunto finito de documentos
- T es un conjunto finito de tareas ($D \cap T = \varnothing$).
- T es una partición tal que $T= T_{(AA)} \cup T_{(AO)} \cup T_{(OO)} \cup T_{(OA)}$. Esto es, en un Grafo de Casos existen cuatro tipos de tareas. $T_{(AA)}$, $T_{(AO)}$, $T_{(OO)}$, $T_{(OA)}$ son disjuntos.
- A es un conjunto de arcos, $A \subseteq ((DxT) \cup (TxD))$
- E: $D \cup T \rightarrow \Sigma^+$ es una función que asocia una etiqueta distinta a cada documento y a cada tarea. Σ^+ es el alfabeto finito de donde se originan las etiquetas.

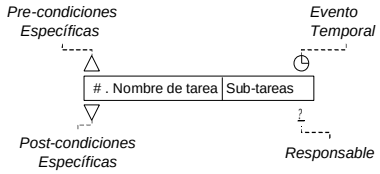
Según la definición 1, existen cuatro tipos diferentes de tareas (AA, OA, AO, OO) cuya representación gráfica se muestra en la figura 2.A. Las tareas son los puntos de transición de los documentos. Cada tarea contiene una especificación de las sub-tareas

que la conforman. Hemos escogido una representación particular que refleja los requisitos para que se ejecute una tarea en un contexto organizacional. Para que se ejecute una tarea no sólo es necesario que existan las entradas requeridas (en este caso los documentos de entrada) sino que puede haber pre-condiciones específicas. Además, la ejecución de una tarea no sólo resulta en la producción de algún documento de salida sino que puede haber post-condiciones específicas. En algunos casos, aún cuando se cuente con las entradas y las pre-condiciones, las tareas requieren que se produzca un evento temporal para ser ejecutadas. Finalmente, cada tarea tiene un responsable directo dentro de la organización. La representación de las tareas se muestra gráficamente en la figura 2.B.

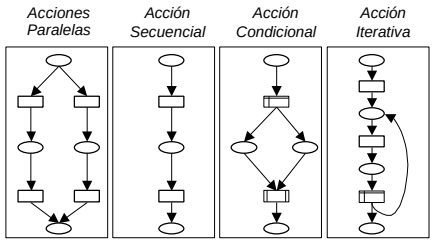
Los diferentes tipos de tareas presentan un comportamiento diferenciado. Una tarea de tipo AA requiere de la presencia de todos sus documentos de entrada y su disparo genera todos sus documentos de salida. El tipo OA se refiere a una tarea que se puede disparar con sólo la presencia de alguno de sus documentos de entrada y su disparo generaría todos sus documentos de salida. El tipo AO es aquella tarea que se dispararía ante la presencia de todos sus documentos de entrada y produciría únicamente a alguno de sus documentos de salida. Y por último, la tarea de tipo OO es la que se puede disparar con sólo la presencia de alguno de sus documentos de entrada y genera sólo alguno de sus documentos de salida.



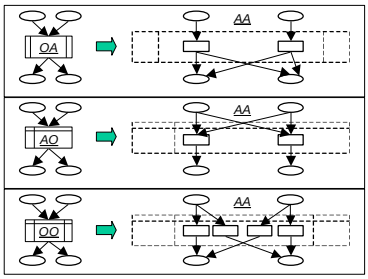
2.A. *Los cuatro tipos de tareas para el modelado de los GC*



2.B. *Representación gráfica de una tarea genérica en un GC*



2.C. *Los cuatro tipos de flujo de acción básicos para el modelado de casos de uso con GCM*



2.D. *Estandarización de las tareas OA, AO y OO de los GCM.*

Fig. 2. Representación de tareas, transformaciones y flujos de acción para modelado de Grafos de Casos

Hemos representado los cuatro tipos básicos de flujo de acción mediante Grafos de Casos. Los flujos de acción se muestran en la figura 2.C. Las acciones secuenciales y las paralelas pueden ser descritas mediante tareas del tipo AA. Sin embargo, las acciones condicionales y las iterativas requieren de la adecuada combinación de los tipos AA, OA, AO y OO.

Para realizar un análisis formal de los casos de uso obtenidos, necesitamos homogenizar el comportamiento de las tareas del Grafo de Casos. Las redes de Petri, que han sido utilizadas para expresar la semántica de los workflows [20], nos pueden proveer el soporte formal para analizar un Grafo de Casos. De esta manera las tareas del Grafo de Casos son visualizadas como las transiciones de la red Petri. Los documentos representan los lugares de la red de Petri. El marcado de la red de Petri podría representar la interacción que muestra la secuencia de estados y no la evolución específica de los casos de uso en conjunto.

Para expresar el Grafo de Casos como una Red de Petri se transforman las tareas OA, AO y OO a tareas del tipo AA. Esto se realiza según se muestra gráficamente en la figura 2.D y según se detalla en la sección 3.5.

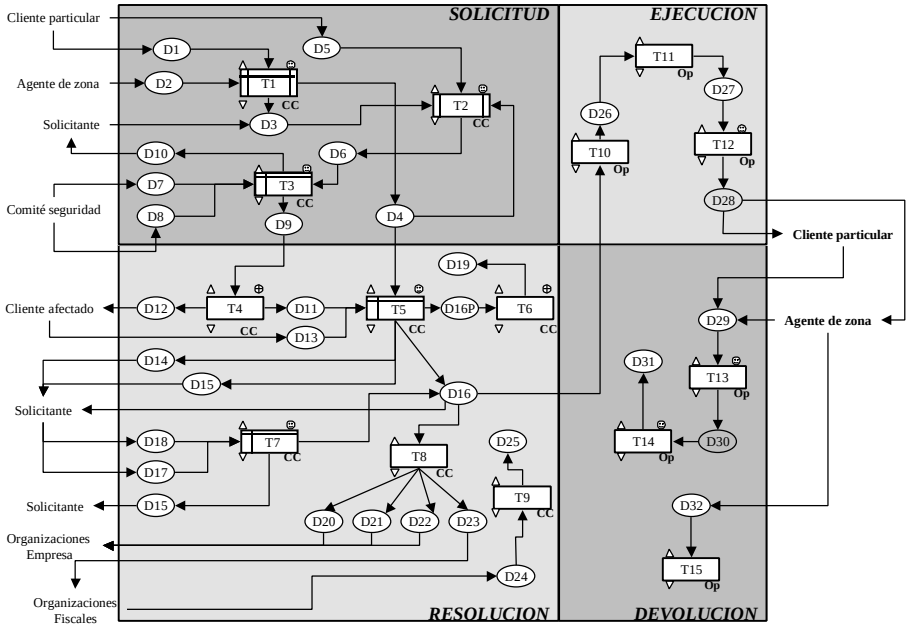


Fig. 3. Grafo de Casos para cuatro procesos en dos secciones de una organización

3.2 Un Ejemplo Real

Para ilustrar la representación de un dDT como un Grafo de Casos nos basamos en un caso real [18], ver figura 3. Hemos tomado cuatro procesos generales de la empresa: *Proceso de Solicitud, Proceso de Resolución, Proceso de Ejecución y Proceso de Devolución*. Dos actores internos son los responsables de las tareas: el Centro de Control (CC), y el Operador Local (Op). El sistema interacciona con siete actores externos: Cliente Particular, Agente de Zona, Solicitante, Comité de Seguridad, Cliente Afectado, Organizaciones de la Empresa y Organizaciones Fiscales.

3.3 Grafos BUC y Grafos UC

El GC es una representación de la funcionalidad del sistema. Por tanto, en él se hayan las formas de interacción entre los usuarios y el sistema. Jacobson [7],[8] distingue dos tipos de interacción de casos de uso: los Bussines Use Cases (BUC) y los Use Cases (UC). Y según el marco de trabajo establecido, a partir del GC se obtienen los BUC y los UC en forma de Grafos BUC y de Grafos UC, respectivamente.

Los Grafos BUC y los Grafos UC reflejan las posibilidades de interacción entre los actores y el sistema. Un Grafo BUC se corresponde con un actor externo. Un Grafo UC se corresponde con un actor interno. Tanto los Grafos BUC como los Grafos UC contienen caminos que puede seguir el flujo de acción del sistema en estudio.

Definición 2. **Camino:** Sea G= (D,T,A,E) un Grafo de Casos, N={n_i \i=1..n, tal que $n_i \in D \cup T$}. Un camino R desde un nodo n_1 a n_k es una secuencia (n_1,n_2,…,n_k) tal que (n_j ,n_{j+1}) $\in$ A, $\forall$ j $\in$ { 1..k-1} .

Esta definición señala que un camino es cualquier secuencia lógica de acción dentro del sistema en estudio. Esta secuencia está compuesta por documentos y tareas los cuales deben estar unidos por arcos dentro del Grafo de Casos.

Las tareas son los puntos de transición de los documentos. Cada tarea tiene asociados dos conjuntos de documentos, los documentos previos y los documentos siguientes. Formalmente se definen estos conjuntos de documentos:

Definición 3. **Documentos Previos y Documentos Siguientes:** Sea G=(D,T,A,E) un Grafo de Casos, el conjunto de documentos previos de la tarea t (t$\in$ T) está definido por ^{o}t={$d_i \in D \mid \exists x \in A$, x conecta d_i con t}. El conjunto de documentos siguientes de la tarea t está definido por t^{o} ={$\forall d_i \in D \mid \exists x \in A$, x conecta t con d_i}.

La consistencia de los Grafos BUC y de los de Grafos UC se puede garantizar si todos sus nodos son alcanzables. Se define un Grafo de Casos fuertemente conectado cuando existe un camino que conecta a cualesquiera dos puntos del grafo:

Definición 4. **Fuertemente conectado:** Sea G= (D,T,A,E) un Grafo de Casos. G es fuertemente conectado sii $\forall$ x$\in$N, $\forall$y $\in$N, N={n_i \i= 1..n, tal que $n_i \in D \cup T$}, existe un camino que conduce desde x hasta y.

De acuerdo con Cockburn [2], los casos de uso describen la forma en la cual los usuarios utilizan el sistema. Nuestros Grafos de Casos deben describir los posibles flujos de interacción usuario y sistema. Para esto definimos:

Definición 5. **Secuencia de Caso:** Sea G= (D,T,A,E) un Grafo de Casos. G es una Secuencia de Caso (SC) sii:

I. D tiene dos documentos especiales i y o. El lugar ***i*** es un lugar fuente, α (i) = 0. El lugar ***o*** es un lugar sumidero, β(o)= 0. Donde α :DxT$\rightarrow$ {1,2,3...} y β :TxD$\rightarrow$ {1,2,3...}

II. Si se agrega una tarea t' a T, la cual conecta los documentos i y o (esto es,{i,t',o} es el camino desde i hasta o), entonces el Grafo de Casos resultante es fuertemente conectado.
III. No existen asociaciones simétricas entre documentos y tareas, es decir, se cumple que, $\forall\ t_i \in T, {}^{o}t_i \cap t_i^{o} = \varnothing$

Ahora podemos definir un Grafo de BUC y un Grafo UC. Ambos tipos de grafos están formados por secuencias de casos. Un Grafo UC es un Grafo de Casos que contiene todas las posibles secuencias de casos para un actor interno dentro de un Grafo BUC.

Definición 6. **Grafo BUC:** Sea G= (D,T,A,E) un Grafo de Casos. G es un Grafo BUC sii contiene todas las posibles secuencias de caso para un documento de entrada de un actor externo.

Definición 7. **Grafo UC:** Sea G= (D,T,A,E) un Grafo de casos. G es un Grafo UC sii:
I. Contiene secuencias de caso que corresponden a un actor interno dentro de un Grafo BUC.
II. Todas la tareas de G son del tipo AA. Esto es, $T= T_{(AA)}$ con lo cual se cumple que $(T_{(AO)} \cup T_{(OO)} \cup T_{(OA)})=\varnothing$.

En la figura 4 se presentan los 12 Grafos BUC que se han identificado (uno por cada documento de entrada) y los 10 Grafos UC (representados por las diferentes regiones sombreadas). Un GC contiene un conjunto de Grafos BUC_i. Cada Grafo BUC_i es un conjunto de Grafos UC_i. Tanto un Grafo BUC_i como un Grafo UC_i consiste de estructuras internas e interfaces externas. Los documentos y las tareas compartidos se consideran parte de las interfaces externas. Las estructuras internas son las mismas que un Grafo de Casos, según las definiciones 1 a la 7.

3.4 El Grafo de Casos Modular

Los documentos y las tareas pueden ser compartidos por diferentes secuencias de caso. Esto ocurre porque un mismo documento puede ir a, o provenir de, diferentes tareas. Por tanto, puede haber intersecciones entre las diferentes secuencias de caso.

Los Grafos BUC pueden compartir documentos y tareas, pero los Grafos UC sólo pueden compartir documentos. Cada Grafo UC recoge el tratamiento de la información aplicable a diferentes secuencias de caso dentro de un Grafo BUC. Dado que el tratamiento de la información se realiza en las tareas, estas no son compartibles entre los Grafos UC.

Los documentos compartidos por los Grafos UC actúan como puntos de conexión entre diferentes grafos de dicho tipo. Esta situación nos ofrece la posibilidad de tratar al Grafo de Casos como una estructura modular según la definición siguiente:

Definición 8. **Grafo de Casos Modular:** Un Grafo de Casos Modular (GCM) es un conjunto $\{G_i = (D_i,T_i,A_i,E_i),\ \backslash i= 1..n\}$ donde:

– Cada G_i es un Grafo UC
– Los T_i deben ser disjuntos para todos los G_i
– Una misma etiqueta no debe ser utilizada para documentos y tareas a la vez, esto es: $\forall G_i , \forall G_j , \neg\exists d \in D_i , \neg \exists t \in T_j$ tal que $E_i (d) = E_j (t)$

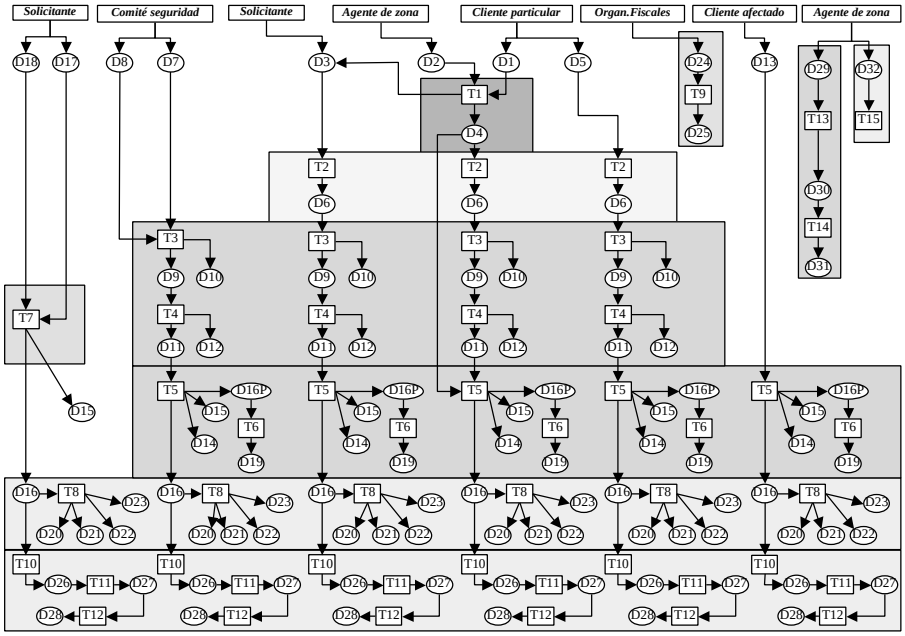


Fig. 4. Secuencias de Casos para los cuatro procesos en las dos secciones de la organización

Para obtener el GCM se debe hacer un análisis del GC para descubrir los bloques de acción comunes a diferentes Grafos BUC. La identificación de estas estructuras comunes permite hacer una factorización del GC conservando intacta la estructura particular de cada uno de los Grafos BUC. La factorización permite proceder a la especificación de las fracciones comunes. Si se expresa la factorización mediante tareas del tipo AA, se satisface la definición de un Grafo UC. Con lo anterior, se garantiza que los casos de uso modelados serán los estrictamente necesarios para especificar la funcionalidad del sistema.

Siguiendo con nuestro ejemplo, obtenemos las estructuras comunes a los diferentes Grafos BUC, lo cual se muestra en la figura 4. Nótese que puede haber Grafos BUC que no comparten estructuras con ninguno otro, como ocurre con los correspondientes a los documentos D24, D29 y D32. Los restantes Grafos BUC sí poseen estructuras comunes. La forma factorizada del GC se aprecia en la figura 5.

Los diferentes bloques de la figura 5 son la base para los Grafos UC pues deben transformarse para contener sólo tareas AA. En el proceso de la transformación de los tipos OA, AO y OO esta implícito un refinamiento de tales tareas mediante la adecuada combinación de tareas del tipo AA. Proponemos que esta labor sea realizada automáticamente. De esta transformación se obtendrá el Grafo de Casos Modular (GCM), del cual el modelo de la figura 5 es su inmediato precursor.

3.5 Refinamiento y Transformación de Tareas

A partir de un GC se pueden extraer automáticamente los casos de uso. Para esto se requiere un proceso algorítmico para el refinamiento y la transformación de tareas que permita que el GC sea expresado como la combinación de Grafos UC.

La generación automática de los Grafos UC debe considerar la especificación de cada tarea, su tipo, su precondición y post-condición. Además se debe conocer la cantidad de documentos de entrada y de salida. Por ejemplo, considérese el cuadro 1 que presenta los datos referentes a las tareas T1 y T2.

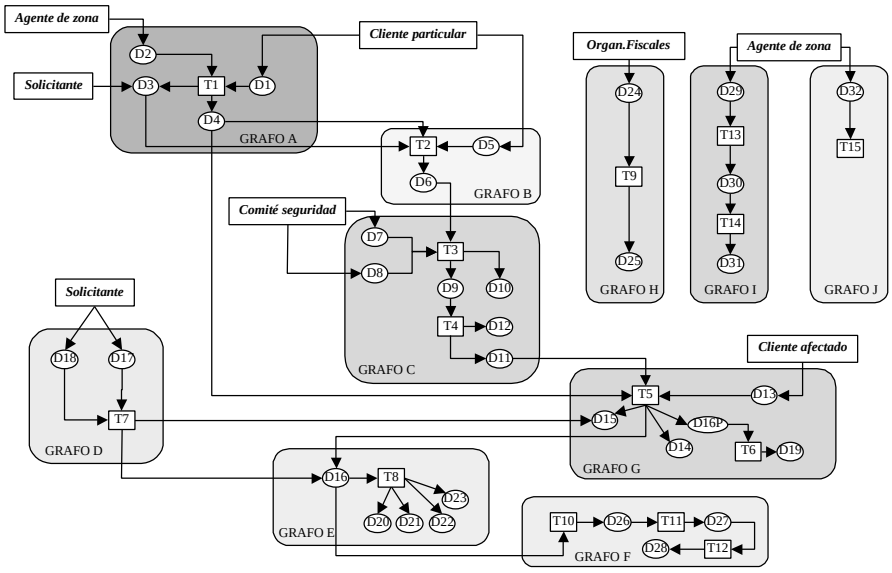


Fig. 5. Factorización de los Grafos BUC para los procesos de la organización

Nombre de Tarea	Detalle	Tipo Tarea	Responsable
T1: Cumplimentar solicitud de descargo	– ***Solicitar*** ingreso de D1 o D2 – ***Verificar*** precondición – ***Procesar*** datos y establecer post-condición – ***Generar*** D3 o D4	OO	Centro de Control
T2: Recibir solicitud de descargo	– ***Solicitar*** ingreso de D3 o D4 o D5 – ***Verificar*** precondición – ***Procesar*** datos y establecer post-condición – ***Generar*** D6	OA	Centro de Control

Cuadro 1. Datos asociados a las tareas T1 y T2 del Grafo de Casos.

Se parte de una abstracción que trata cada tarea como compuesta genéricamente por una sucesión de cuatro sub-tareas: *Solicitar*, *Verificar*, *Procesar y Generar*. A partir de una visión genérica de las tareas, se puede refinar cada una de ellas según la secuencia de sub-tareas. Se deben realizar las transformaciones acordes con el patrón

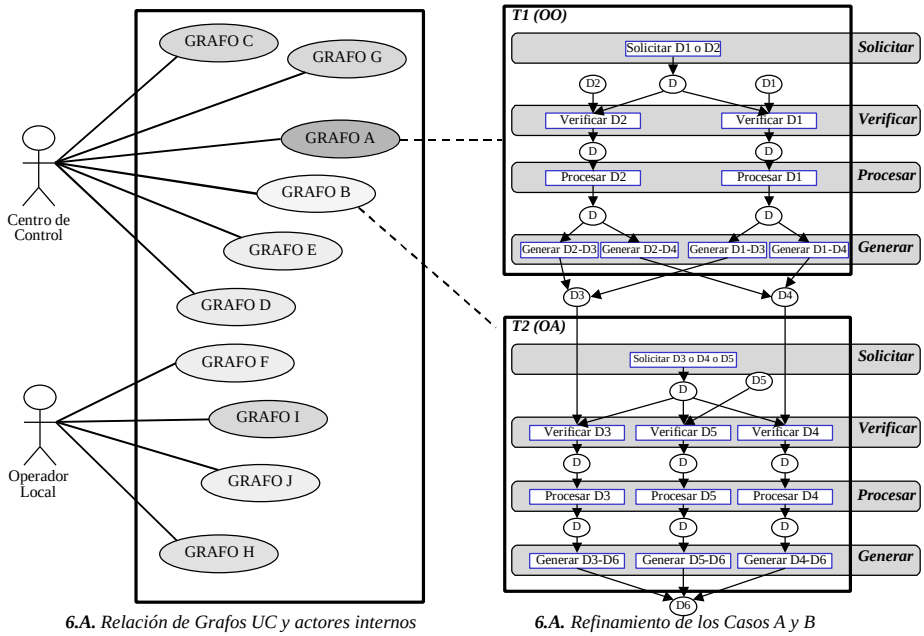
dado en la figura 2.D. Procediendo de esta manera con cada tarea, el Grafo de Casos quedará expresado en términos del estándar AA. Finalmente, cada Grafo UC correspondería a una red de Petri y también todo el GCM sería una red de Petri. Las labores de validación y de verificación determinadas por el marco de trabajo permitirían realizar los ajustes necesarios en las tareas.

La sub-tarea *Solicitar* es, por defecto, realizada por el sistema (aunque se puede indicar lo contrario). Esta sub-tarea produce un documento interno que representa la respuesta o ejecución de la acción correspondiente a lo solicitado.

El comportamiento de la sub-tarea *Verificar* es diferenciado según las entradas y según sea el tipo de tarea que se desee refinar o transformar. Para el caso de tareas OA y OO, la sub-tarea verificar se debe descomponer en tantas sub-tareas como entradas existan. Para las tareas de tipo AO y AA, la sub-tarea Verificar no se subdivide. La acción de Verificar se asocia con la precondición de la tarea.

La sub-tarea *Procesar* se encarga de establecer la post-condición y se fracciona automáticamente en una cantidad de sub-tareas igual a la que corresponde a la sub-tarea Verificar que le precede.

La sub-tarea *Generar* también posee un comportamiento dependiente del tipo de tarea madre. Para las tareas OA y AA, la sub-tarea Generar se fracciona en un número de sub-tareas igual al número de fracciones de la sub-tarea Procesar. De cada una de esas sub-tareas Generar saldrá un arco para cada uno de los documentos de salida de la tarea madre. Para las tareas AO y OO, la sub-tarea Generar debe ser subdividida de modo que por cada sub-tarea Procesar existan tantas sub-tareas Generar como salidas tenga la tarea madre. Cada sub-tarea Generar que corresponda a cada una de las sub-tareas Procesar tendrá un arco que le conecta con una salida materna diferente.



6.A. *Relación de Grafos UC y actores internos* ***6.A.*** *Refinamiento de los Casos A y B*

Fig. 6. Relación de los Grafos UC con los actores y refinamiento de los grafos de casos A y B.

3.6 Obtención de los Casos de Uso

De acuerdo con la definición 6, los BUC se obtienen como un grafo que contiene todas las secuencias de caso originadas de un documento proveniente de un actor externo. Según la definición 7, para obtener los casos de uso es necesario transformar las tareas OA, AO, y OO a tareas AA. Con las tareas refinadas y transformadas, la expresión de los casos de uso en formato de plantillas es una labor que consiste en hacer un seguimiento del marcado de la red de Petri. Deben determinarse algunos estándares para nombres de documentos intermedios, y para la nomenclatura de los casos para expresarlos en forma de plantilla, como la propuesta en Durán [4].

La figura 6.A muestra las relaciones entre los diferentes Grafos UC obtenidos y los actores internos. La figura 6.B muestra los casos A y B que contienen a las tareas T2 y T3 transformadas según lo expuesto en la sección 3.5.

Los casos de uso expresados mediante plantillas, o mediante una red de Petri o su traducción al formato requerido, pueden ser enviados a un gestor de mecanos para finalmente ajustar los assets bajo la semántica del repositorio y así poder recuperar los mecanos o formarlos en tiempo de reutilización, de acuerdo con García [5].La expresión de casos de uso basada en redes de Petri debe permitir la comparación y la adaptación de los requisitos. La definición de los mecanismos de adaptación y de comparación serán objeto de nuestro trabajo futuro.

4 Conclusiones y Tareas Inmediatas

Con el presente trabajo se ha propuesto la derivación de los casos de uso a partir de un workflow que modela el flujo de información en un dominio y en el contexto de la reutilización del software. Se ha propuesto la normalización del proceso de captura de los requisitos de usuario mediante un Grafo de Casos y una red de Petri. Los GCM propuestos son una alternativa para modelar sin ambigüedad la dinámica del negocio con la información mínima requerida. Los GCM representan los flujos de actividad mediante un esquema de control que formaliza la captura de requisitos y permite la escalabilidad y la trazabilidad.

Con la funcionalidad capturada se podrán realizar análisis diversos que permitan establecer familias de casos de uso para diferentes propósitos. Las familias de casos de uso deberán permitir la comparación y la adaptación. Además, las familias de casos de uso podrán ser la base para efectuar estimaciones de coste para implementación de los diferentes conjuntos de casos de uso.

Los GCM deberán ser probados y validados en la definición de los requisitos de usuario. No obstante, nosotros esperamos que su funcionalidad paralela con el dominio del negocio y su fundamento en la teoría de redes de Petri, sean elementos válidos para incrementar rendimientos en el desarrollo de software con reutilización.

Nuestras tareas inmediatas consisten en proporcionar:

- Una herramienta para la derivación automática de los casos de uso desde los Grafos de Casos.
- Un modelo de comparación de requisitos para desarrollo con reutilización a partir de los Grafos de Casos.

Referencias

[1] Booch, G., Rumbaugh, J., Jacobson, I., "El lenguage unificado de modelado", Adisson Wesley, 1999.

[2] Cockburn, A. Writing Effective Use Cases. Humans and Technology in preparation for Addison-Wesley Longman, Q3 2000. 1999.

[3] Collongues, Huges and Laroche. "Merise. Methode de conception", Dunod, 1987.

[4] Durán, A. et al., "Una propuesta metodológica para la recolección de requisitos de un sistema software", III Jornadas de Trabajo MENHIR, 1998.

[5] García, F.J., Modelo de Reutilización Soportado por Estructuras Complejas de Reutilización Denominadas Mecanos. TESIS DOCTORAL, Universidad de Salamanca, 2000.

[6] IEEE, "Recommended Practice for Software Requirements Specifications", Std 830-1993

[7] Jacobson, I., et al., Object-Oriented Software Engineering – A use case driven approach. Addison-Wesley, 1992.

[8] Jacobson, I., Griss, M., Jonsson, P., Software Reuse, Architecture, Process and Organization for Business Success. ACM, 1997

[9] Karlsson, E., Software Reuse: A Holistic Approach. John Wiley & Sons. 1995 – ISBN 0-41-95819-0

[10] Kotonya, G.; Sommerville, I., Requeriments Engineering: Processes and Techniques. USA, Willey. 1997.

[11] Laguna, M.A., Marqués, J.M., García, F.J., "A user requirements elicitation tool". IV Jornadas de Trabajo MENHIR, Sedano, 1999.

[12] Lee, W.J., Cha, S.D., Kwon, Y.R., "Integration and Analysis of Use Cases Using Modular Petri Nets in Requirement Engineering", IEEE Transactions on software engineering, 24:12, diciembre 1998.

[13] López, O; Laguna, M.A.; Marqués, J.M., Normalización de Assets de Requisitos en el Contexto de la Reutilización Sistemática del Software. V Jornadas de Trabajo MENHIR, Granada, Marzo 2000.

[14] Pohl K., Requeriments Engineering: An Overview. Informatik V. RWTH, Germany. 1997.

[15] Pressman, R., "Ingeniería del Software", 4ta. Edición, Madrid, McGraw-Hill, 1998.

[16] Prieto-Díaz, R., Classification of reusable modules. In Software Reusability. Concepts and Models. Volume 1 of Frontier Series, edited by Biggerstaff and Perlis, 1989. ACM Press. Pages 99-122.

[17] Rational Software, "A Rational Aproach to Software Development Using Rational Rose 4.0", 1998.

[18] Sánchez, M.B., Gestión de Intervenciones en red de IBERDROLA. Proyecto de fin de carrera. Escuela Universitaria Politécnica. Universidad de Valladolid, 1999.

[19] Silva, M., "Las redes de Petri en la automática y la informática", Madrid, Editorial AC, 1985.

[20] Van der Aalst, W.M.P., "The application of Petri Nets to Workflow Managment", Eindhoven, University of Technology, Netherland, 1997. **URL:** wwwis.win.tue.nl/~wsinwa/jcsc/jcsc.html

[21] WFMC. Workflow Management Coalition Terminology and Glosary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996. http://www.aiim.org/wfmc/standards/docs/glossy3.pdf