Reutilización de Requisitos en el Modelo Mecano

Oscar López
Instituto Tecnológico de Costa Rica
olopez@infor.uva.es

Miguel Ángel Laguna
Universidad de Valladolid
mlaguna@infor.uva.es

Francisco José García
Universidad de Salamanca
fgarcia@gugu.usal.es

Resumen

La reutilización de requisitos amplía el espectro de posibilidades en reutilización del software. El desarrollo con reutilización se basa en satisfacer requisitos que en sí mismos son elementos reutilizables. Los mecanos proporcionan un soporte estructural para relacionar requisitos, diseño e implementación. Sobre esa base, este artículo propone un proceso para derivar elementos reutilizables desde las etapas iniciales del ciclo de vida. El punto de partida son requisitos representados, clasificados, y almacenados en mecanos. Estos requisitos deber ser comparados y adaptados. Más que una propuesta de clasificación de assets, el artículo plantea una estrategia para reutilizar software desde los requisitos funcionales, la cual es una aproximación para acelerar el desarrollo de soluciones con reutilización de software.

Palabras Clave. Reutilización del software, ingeniería de requisitos, casos de uso, requisitos genéricos.

1 Introducción

La reutilización de los requisitos es un enfoque importante para el éxito en el desarrollo con reutilización. El software incluye elementos tanto de implementación como de las fases de diseño y de análisis (como diseños, documentación, requisitos, arquitecturas e interfaces) que son potencialmente reutilizables [12]. Los requisitos reflejan conocimiento respecto a un dominio [5] lo cual es valioso en la reutilización ideal que trata de reutilizar el conocimiento [18]. Además, dado que la reutilización de los elementos de diseño e implementación obedece a la satisfacción de una serie de requisitos, el punto de inicio de una estrategia de reutilización sistemática son los requisitos, en particular los funcionales [19]. Por todo lo anterior, la reutilización de requisitos es una alternativa para el mejor aprovechamiento del esfuerzo de desarrollo.

Los requisitos pueden reutilizarse con enfoques distintos como puede ser: (a) En desarrollo de nuevas especificaciones, (b) En mantenimiento de las aplicaciones existentes, (c) En desarrollo de nuevas aplicaciones [20]. Independientemente del enfoque que se adopte, la reutilización sistemática de los requisitos pasa por definir formas aptas de representar y almacenar las especificaciones en una fase de desarrollo para reutilización, y por definir la forma de comparar y adaptar los elementos reutilizables en una fase de desarrollo con reutilización. En síntesis, la reutilización de los requisitos es análoga a la reutilización del software por lo que requiere formas de representar, almacenar, comparar y adaptar los elementos potencialmente reutilizables.

La representación y almacenamiento de los requisitos para reutilización determina la necesidad de normalizar los assets. La representación mediante escenarios, casos de uso y metas [14] incluye formas diversas por lo que se impone una labor de normalización para su almacenamiento como assets en el repositorio. Con redes de Petri se puede normalizar los requisitos para ser almacenados y reutilizados [15].

Aún con los requisitos normalizados, hay un inconveniente para la comparación en la fase del desarrollo con reutilización. La comparación de escenarios puede no aportar soluciones eficientes para decidir si un asset es aplicable en un contexto diferente al de

su origen pues los escenarios son instancias de problema. Se requieren dos acciones básicas para poder comparar requisitos (ver figura 1). La primera es definir un proceso de abstracción de los escenarios del problema que se intenta abordar con reutilización y obtener un modelo simplificado. La segunda es combinar criterios de generalización y genericidad para gestionar los assets de requisitos. La simplificación de los requisitos se logra mediante el análisis por reducción de redes de Petri [21]. Para el tratamiento de assets mediante criterios de genericidad se define un modelo de requisitos genéricos que sea compatible con el esquema de clasificación de assets. Los requisitos genéricos relacionan requisitos funcionales (casos de uso genéricos) y modelos de objetos de sistemas (OSM) [23]. La genericidad de los assets actúa como elemento de comparación y adaptación, mientras que la generalidad sirve para la descripción y la recuperación.

En el presente artículo se propone una aproximación para reutilizar requisitos cuyo

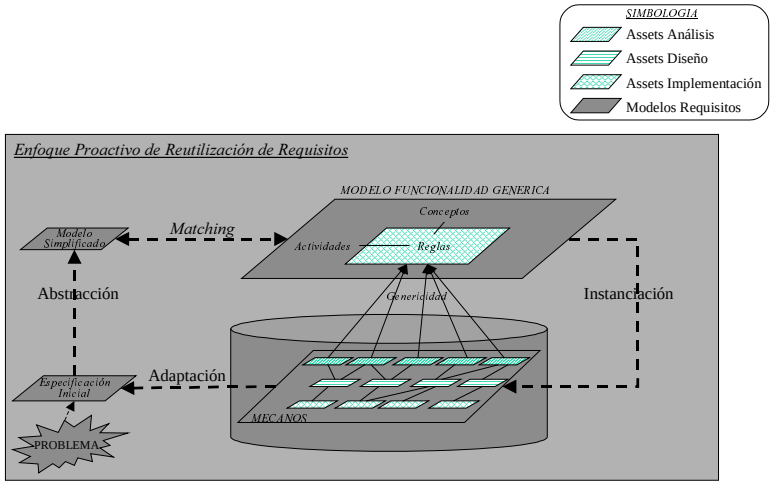

Fig. 1. Reutilización del conocimiento del dominio en el desarrollo de software: El modelo simplificado del problema es la entrada del matching con modelos genéricos para instanciar los elementos adaptables al problema dado.

objetivo es favorecer la reutilización entre dominios desde las etapas iniciales del desarrollo. Para ello se cuenta con requisitos relacionados con modelos de diseño e implementación en mecanos. Sobre la base de un mecanismo de abstracción de requisitos y la existencia de assets que pertenecen a dominios específicos y que se corresponden con modelos genéricos, proponemos la forma de comparar e instanciar assets. La comparación se realiza mediante analogía basada en escenarios y metas [17][16] con un proceso automático para decidir si un modelo de requisitos es reutilizable en un problema dado. La determinación de los elementos de mecanos aptos para el problema permite seleccionar una instancia de requisitos del repositorio o instanciar un modelo genérico de requisitos.

El artículo incluye tres secciones adicionales. La sección 2 presenta el modelado de dominios y su integración con modelos genéricos. La sección 3 presenta el proceso de reutilización del conocimiento del dominio en desarrollo con reutilización. La sección 4 incluye conclusiones y trabajo inmediato.

2 Modelado de Dominios con Requisitos Genéricos

La reutilización introduce la ingeniería de dominios y la ingeniería de aplicaciones como dos procesos concurrentes [12]. Un dominio es un área funcional diferenciable, con requisitos y rasgos (*features*) similares, que puede ser soportada por algún tipo de sistema software. Mediante la ingeniería de dominios se captura, organiza y representa la información útil para el desarrollo de sistemas software de manera que esta pueda ser reutilizada para crear nuevos sistemas software [18].

Para organizar y aprovechar la información del dominio se debe contar con procesos y estructuras adecuados [4]. En cuanto al proceso, se deben comparar (*matching)* el modelo del problema y los modelos reutilizables para seleccionar y adaptar los elementos precisos [9]. Desde el punto de vista de las estructuras, se requiere del soporte para la representación, organización y recuperación de los assets [7]. En esta sección proponemos la integración de dominios y modelos genéricos de requisitos en mecanos como soporte estructural para la reutilización entre dominios.

2.1 El modelado de requisitos en los dominios

Los requisitos se pueden derivar mediante un enfoque multi-mundos según el marco definido por las teorías de representación del conocimiento, las teorías de dominios, y las teorías de procesos [11]. El enfoque multi-mundos concibe el modelado del negocio (*enterprise modelling*) como una forma de desglosar y de modelar el mundo real [2], ver figura 2. Del modelo del negocio se obtienen los modelos de requisitos que luego se relacionan con los respectivos modelos de sistemas de información. En el modelado de requisitos debe reflejar los contenidos e interacciones de cuatro sub-modelos del negocio: *Objetivos*, *Conceptos*, *Actores, Uso y Actividades*.

Los dominios se pueden ver como un esquema de organización de los requisitos cuya utilidad es apreciable en entornos de reutilización sistemática del software [4]. Un dominio es un subconjunto de conocimiento por lo que se puede hablar del conocimiento propio de cada dominio. El conocimiento se puede representar mediante asociaciones de conceptos, actividades y reglas. En el plano de la ingeniería del software, el analista traduce el conocimiento del dominio en una serie de requisitos funcionales y no funcionales. Los requisitos, especialmente los funcionales [19], juegan un rol fundamental para la selección de elementos reutilizables. La organización de requisitos bajo un esquema de integración de dominios permite identificar las oportunidades de reutilización del software en un contexto distinto al de su origen.

2.2 Dominios, subdominios y dominios genéricos

Los requisitos representan el conocimiento de un dominio particular por lo que se deben tener criterios de clasificación de assets para reflejar el conocimiento de distintos niveles de dominio. Un dominio es cualquier área funcional diferenciable dentro de un contexto [18], pero esta definición engendra falta de consenso respecto a la delimitación de dominios. Es preciso definir los niveles de dominio, subdominio y dominio genérico.

Se puede definir un dominio particular sobre cualesquiera conjuntos de conceptos, reglas y actividades de interés. Con esta definición, se puede establecer que existe el dominio particular de automóviles, o el dominio particular de telefonía móvil, por

ejemplo. No obstante, la complejidad de esos dominios particulares obliga a dividir el contexto en subdominios. Un subconjunto de los elementos de un dominio se llama un subdominio. Así se puede determinar el subdominio de ventas, o el de alquiler, dentro del dominio particular de automóviles, por ejemplo. Un subdominio se define sobre un subconjunto de actividades de un dominio particular de interés:

La clasificación de assets por subdominios favorece la reutilización intradominios pero no interdominios. Si en el repositorio de elementos reutilizables sólo se incluyen assets de un dominio particular, por ejemplo automóviles, cualquier intento de reutilización soportado en ese repositorio se relaciona de manera directa con un único dominio y sus subdominios. Sin embargo, en un enfoque de reutilización más ambicioso se intenta mantener assets que pertenecen a distintos dominios particulares dentro del repositorio. Se puede crear una estructura de organización en la que cada dominio se representa de manera independiente en el repositorio. Esta organización presenta el inconveniente de requerir un modelo específico para cada dominio particular lo cual es crítico y costoso y no facilita la reutilización entre dominios.

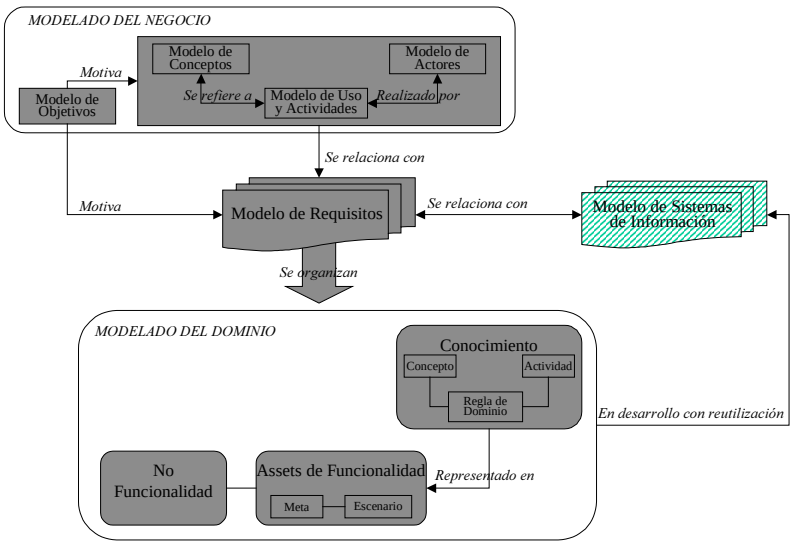

Fig. 2. Modelado del Negocio, Modelado de Requisitos y Modelado de Dominios. Un dominio se representa mediante conocimiento, requisitos funcionales y no funcionales para abordar el desarrollo de sistemas con reutilización del software.

Para clasificar requisitos reutilizables se requiere un nivel de mayor abstracción respecto a los dominios particulares y los subdominios que permita mantener la especificidad de cada dominio al mismo tiempo que hace visible la generalidad entre los dominios y subdominios. Ese mayor nivel de abstracción se tiene en un dominio genérico que permite relacionar instancias de dominios particulares distintos lo cual es ventajoso para la reutilización entre dominios particulares. Análogo a un dominio particular, un dominio genérico es un conjunto de actividades genéricas, conceptos genéricos y reglas genéricas relacionados y que además posee subdominios genéricos.

La abstracción de la generalidad sobre dominios particulares induce a pensar en dominios genéricos (banca genérica, seguros genéricos, etc.) o en subdominios genéricos (contabilidad genérica, venta genérica, etc.). Estos parecen más aplicables a

entornos de generación automática de software que para reutilización orientada a la adaptación de assets. Además, la abstracción de dominios y subdominios genéricos demanda la construcción de un modelo de conocimiento en cada nivel genérico, lo cual es complejo. Para articular soluciones software mediante adaptación, como es el caso de los mecanos [8], es necesario un dominio genérico más restringido para reducir el costo de la adaptación [1]. Este nivel de genericidad debe integrarse en el esquema de clasificación para permitir la selección de assets particulares mediante una proyección desde los dominios particulares. La reutilización de mecanos requiere un criterio de abstracción y de representación diferente a dominios o subdominios genéricos.

2.3 Representación de dominios en Mecanos

Los modelos de dominios se han propuesto como alternativa para clasificar los elementos reutilizables e identificar la oportunidad de reutilización de manera eficiente [18]. No obstante, la forma de integrar las abstracciones de dominio en mecanos debe estar acorde con el enfoque de reutilización de assets mediante adaptación.

La reutilización a partir de requisitos funcionales exige clasificar los assets con un criterio distinto a dominios genéricos. Es decir, se debe definir un criterio de clasificación extra e independiente del esquema de clasificación (por ejemplo facetas [18], descriptores funcionales [8]). Este criterio puede establecerse en términos de modelos genéricos de funcionalidad. Reubenstein [20] propuso usar modelos genéricos para evitar la construcción de modelos específicos de conocimiento para cada área de aplicación. Sutcliffe y Maiden [23] han propuesto una serie de OSM (suministro, control, sensores, etc) que permiten extender el concepto del conocimiento genérico más allá de los entornos de diseño orientado a dominios específicos. Se han descrito siete familias de casos de uso genéricos: diagnóstico, búsquedas de información, reservación, confrontación, calendarización, planeación, análisis-modelado [22]. Esos trabajos pueden integrarse en el esquema de clasificación de assets para la reutilización del conocimiento inter-dominios (entre dominios funcionales distintos, como alquiler de vehículos y préstamo de libros).

Los ingenieros de software desarrollan estructuras de conocimiento para reconocer las similitudes entre problemas [23]. Por lo anterior, en el repositorio deben existir modelos abstractos apropiados (modelos de requisitos genéricos), que guarden una relación de genericidad con las instancias de assets de funcionalidad de dominios particulares. Esta genericidad proporciona el criterio para la reutilización de los assets que se hallan clasificados según criterios de generalidad de dominios. De esta manera, la organización de los assets permite tanto la reutilización intra-dominios (se pueden recuperar los assets de un dominio dado) como inter-dominios (se pueden reutilizar elementos que coincidan en su dimensión genérica aunque sean de dominios distintos).

En la figura 3 se presenta el modelo de requisitos genéricos y sus relaciones con los assets de dominios particulares. Los casos de uso de dominios particulares son instancias de actividades genéricas, las metas son instancias de reglas de dominio y los actores son instancias de conceptos. Los conceptos no se expresan directamente en los requisitos funcionales pero se pueden ver implícitos en las metas. Los OSM están implícitos en el modelo de requisito genérico como conjuntos de objetos cooperantes donde la abstracción se ubica en las relaciones entre los conceptos.

Para reutilizar assets de diseño e implementación, teniendo como punto de partida los assets de funcionalidad asociados en mecanos, se puede interactuar con el repositorio y sus servicios de biblioteca de reutilización. En concreto, se pueden expresar consultas guiadas por los requisitos funcionales para recuperar los assets de interés desde el repositorio. Las consultas se pueden realizar en términos de (a) dominio (subdominio) particular, (b) actividades genéricas, o (c) conceptos genéricos (OSM). Además de la posibilidad de realizar esas consultas al repositorio, la existencia de modelos de requisitos genéricos dentro del repositorio permite la composición de soluciones mediante assets lo cual será parte de nuestro trabajo futuro.

La representación de los dominios particulares por medio de requisitos en los mecanos se manifiesta en el esquema de clasificación de los assets de funcionalidad (figura 3). Cada instancia de asset pertenece a un dominio (subdominio) particular. El conjunto de todas las instancias de assets de funcionalidad del repositorio constituye el universo del discurso, donde coexisten varios dominios particulares. El universo de dominios particulares se divide en un conjunto común de subdominios. De esta manera, cada instancia de assets de funcionalidad debe corresponder a un subdominio (y un dominio) y a la vez a un requisito genérico.

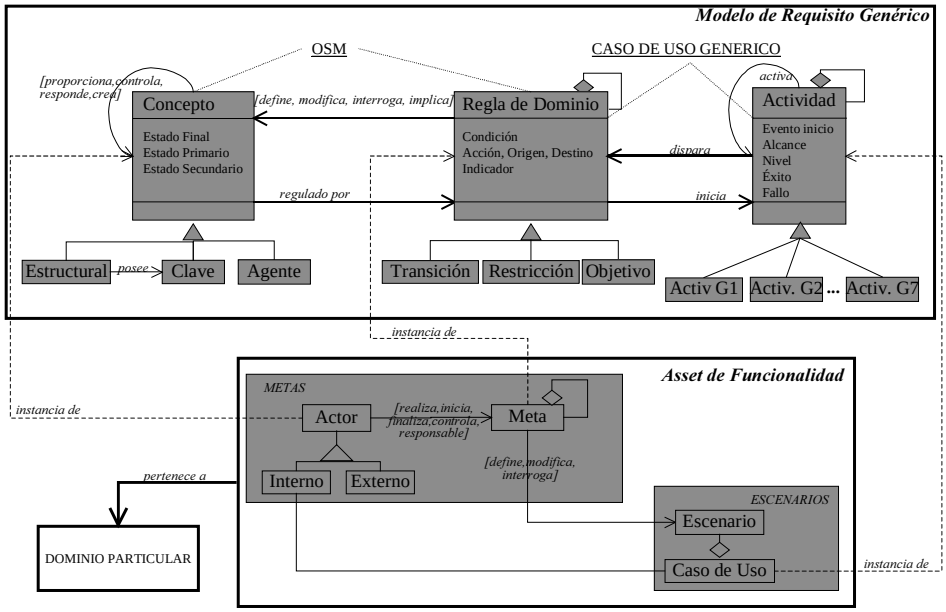

Fig. 3. Modelado de requisitos genéricos y assets de funcionalidad. Los assets pertenecen a dominios particulares. El modelo de requisitos genéricos combina OSM y casos de uso genéricos.

3 Proceso para reutilizar requisitos de dominio

El soporte estructural dado por la organización de assets por dominios y por requisitos genéricos en mecanos nos permite proponer el proceso para reutilizar los requisitos de dominio. El conocimiento presente en assets de requisitos se puede reutilizar en desarrollo con reutilización. La dimensión de genericidad hace corresponder a los assets con un modelo genérico de actividades y un modelo genérico de conceptos a la vez, figura 3. Por su parte, el problema que se intente abordar con

reutilización pertenece a un dominio específico el cual puede no estar representado en el repositorio, lo cual no significa inexistencia de elementos reutilizables para el problema. La doble dimensión genérica permite recuperar assets análogos al problema, ya sea desde la óptica de modelos genéricos de actividades o modelos genéricos de conceptos.

Se supone que el desarrollador con reutilización reconoce el dominio de su interés para interactuar con el repositorio. Si el dominio de interés se encuentra en el listado de disponible en el repositorio, el reutilizador puede especificar una consulta a través de los servicios de biblioteca de reutilización y seleccionar los assets necesarios con base en su experiencia propia. Si no se dispone de assets específicos para el problema (por ejemplo, el desarrollador buscar assets para el problema de venta de libros pero el repositorio posee assets para venta de autos) se puede obtener elementos reutilizables mediante un proceso de reutilización del conocimiento de dominios según se presenta en la figura 1 y el cual incluye los siguientes pasos:

1. Abstracción del problema para llegar a un modelo simplificado
2. Comparación *(matching)* del modelo simplificado y los modelos genéricos
3. Instanciación del modelo genérico adecuado al problema a resolver
4. Adaptación del modelo instanciado para satisfacer los requisitos del problema

La abstracción del problema para llegar al modelo simplificado se puede realizar automáticamente según se detalla en [15] para generar los casos de uso desde una descripción de la forma de trabajo de los usuarios. También se puede expresar el modelo del problema en términos del lenguaje de consultas del repositorio. El *matching* (comparación) de los modelos del problema y los assets ofrecidos por el repositorio se basa en el trabajo descrito en [16] y consiste en comparar metas de usuario y escenarios. Esta comparación de modelos es el paso previo a la instanciación.

3.1 Comparación del problema y los assets del repositorio

La *dimensión de genericidad* presente en los assets de requisitos permite dos vías de recuperación de elementos para el matching con el modelo del problema. Dado que los requisitos funcionales se especifican con base en casos de uso, y estos se corresponden con actividades de dominio, una forma natural de iniciar la búsqueda de elementos reutilizables es por la dimensión genérica de actividades (reservación, diagnóstico, comparación, etc) o por dominios particulares para orientar la recuperación de assets. Así por ejemplo, se puede expresar la consulta en términos de *venta de libros* con lo cual puede suceder que el repositorio ofrezca un modelo de *venta de servicios* y uno de *venta de pólizas*. Si la consulta no encontrara assets disponibles, entonces el proceso de recuperación se orienta por los modelos genéricos de conceptos (OSM). Por ejemplo, puede suceder que no exista ningún modelo específico de ventas pero si que exista un modelo genérico de suministro que incluye conceptos donde se transfieren bienes o servicios entre conceptos. En resumen, con la dimensión de genericidad se garantiza que, en el ámbito del conocimiento representado en el repositorio mediante assets de requisitos, se puede ofrecer al reutilizador algún modelo que pueda ser instanciado.

Para ilustrar la comparación de los modelos de requisitos, en la figura 4.A se presentan los escenarios y las metas del problema de venta de libros. La red de

escenarios se puede obtener automáticamente a partir del workflow tal y como se describe en [15]. La integración de las metas en la red de escenarios se fundamenta en las relaciones que existen entre los casos de uso y las metas [3][6]. Supóngase que para el dominio de venta de libros no existen assets específicos en el repositorio. Supongamos además que en el repositorio sí que existen dos modelos de ventas, uno de venta de servicios (figura 4.B) y otro de venta de pólizas (figura 4.C).

Mediante el *matching* (comparación) de modelos se debe determinar si los modelos disponibles en el repositorio son potencialmente reutilizables en el problema de la venta de libros. La comparación de los modelos se realiza mediante los siguientes pasos:

a. **Inclusión de metas:** Se preseleccionan las modelos del dominio cuya jerarquía de metas incluya (inclusión de nodos) a las metas raíz y las metas hoja de la jerarquía de metas del problema. Esto supone que las metas de la instancia y del problema están expresadas mediante un lenguaje común.

b. **Satisfacción de metas:** Del subconjunto obtenido en el punto a) se seleccionan aquellos modelos del dominio cuya estructura interna de metas (la jerarquía de metas de la instancia del dominio sin considerar los nodos raíz) satisface el logro de las metas hoja del problema. Esta condición asegura que en la instancia del dominio existe al menos un camino lógico para satisfacer cada una de las metas finales del problema

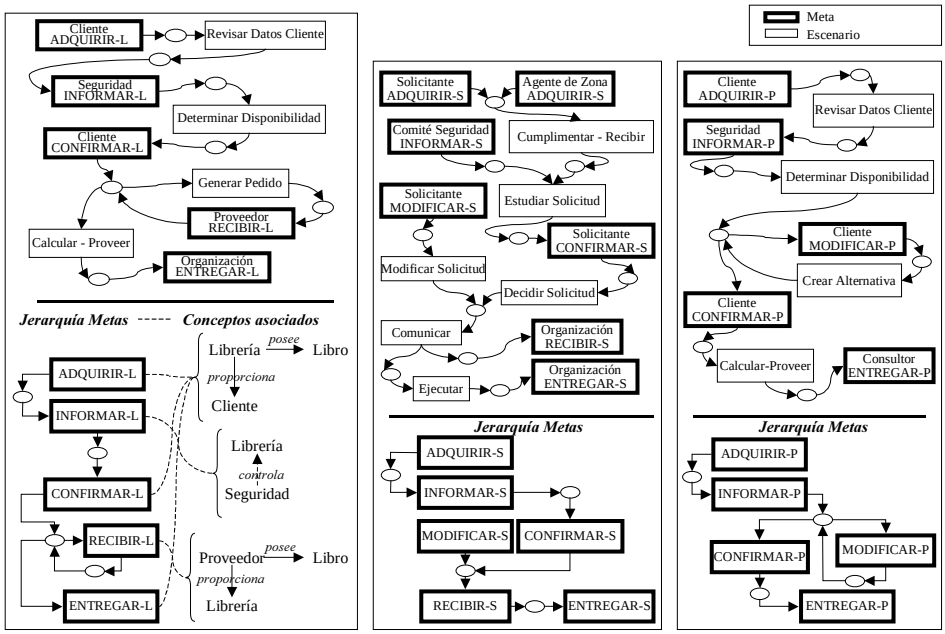

Fig. 4.A. Problema de venta de libros 4.B. Instancia de proceso de órdenes 4.C. Instancia venta de pólizas

Fig. 4. Modelos del problema de venta de libros, del proceso de órdenes y de venta de pólizas

De la comparación se deduce que ambos modelos alternativos incluyen las metas hoja y las metas raíz del problema. La meta hoja del problema es ENTREGAR-L que se corresponde con ENTREGAR-P y ENTREGAR-S. La meta raíz del problema es ADQUIRIR-L que se corresponde con ADQUIRIR-P y ADQUIRIR-S. Por otra parte, la estructura de metas de ambos modelos satisfacen el logro de metas hoja del problema.

Se demuestra mediante refutación tal y como se ilustra en [16]. En conclusión ambos modelos alternativos son potencialmente reutilizables en el problema planteado.

3.2 Instanciación del modelo genérico de actividades

La comparación permite determinar si los modelos ofrecidos se pueden asociar con el problema. De los modelos potencialmente reutilizables para el problema dado, se debe ofrecer al desarrollador el más adecuado en función del menor costo esperado de adaptación. Al proceso de seleccionar el modelo más adecuado se le denomina instanciación del modelo del dominio. Si dentro del repositorio no existieran instancias específicas aplicables al problema se puede instanciar el modelo genérico correspondiente. Es decir, la instanciación es un proceso que permite obtener una instancia de dominio ya sea por discriminación de entre las disponibles como assets de requisitos o instanciando el modelo genérico correspondiente.

3.2.1 Discriminación de instancias

La discriminación de entre los modelos disponibles se basa en los siguientes pasos:

c. **Secuencias análogas de escenarios:** Del subconjunto obtenido en el punto b) se seleccionan la instancia del dominio que posea la mayor cantidad de secuencias de escenarios de instancia que sean aplicables al problema. Una secuencia de escenarios de instancia es aplicable al problema si y sólo si:

- Parte de un nodo raíz de la instancia y satisface una meta hoja de la instancia
- Satisface una meta raíz del problema y una meta hoja del problema

d. **Actividades:** En caso de que el subconjunto obtenido en el punto c) posea más de un elemento, se debe seleccionar la instancia cuyos escenarios sean los más idóneos.

El modelo de venta pólizas posee dos secuencias de escenarios análogas al problema: (1) Revisar Datos Cliente – Determinar Disponibilidad – Calcular Proveer, y (2) Revisar Datos Cliente – Determinar Disponibilidad – Crear Alternativa – Calcular Proveer. Por su parte, el modelo de procesado órdenes sólo presenta una secuencias de escenarios análoga al problema: Cumplimentar Recibir – Estudiar Solicitud – Decidir Solictud – Comunicar – Ejecutar. En consecuencia, el modelo 4.B se selecciona como el más adaptable al modelo del problema.

3.2.2 Instanciación de actividades genéricas

Para instanciar actividades genéricas se tienen dos opciones. El desarrollador determina mediante un lenguaje de consultas qué actividades desea instanciar o el repositorio determina las actividades correspondientes a partir de un modelo del problema (figura 1). El desarrollador con reutilización puede expresar su consulta al repositorio ya sea en términos de dominio específico o en términos de actividades genéricas. En esta situación el repositorio, a través de sus servicios de biblioteca de reutilización, seleccionará un conjunto de actividades genéricas las cuales el desarrollador completará con los parámetros correspondientes. Para determinar automáticamente las actividades genéricas que se corresponden con un problema se requiere un proceso selección de actividades genéricas y de instanciación de los parámetros de las actividades seleccionadas.

La selección de actividades genéricas correspondientes a un problema se realiza mediante un recorrido por la estructura de actividades genéricas. Del problema que se pretende abordar con reutilización se debe obtener un modelo simplificado que reúne la funcionalidad esencial. Ese modelo se expresa como una red de escenarios y metas de donde se abstrae una jerarquía de metas del problema. Por su parte, los casos de uso genéricos se modelan como actividades genéricas asociadas con reglas de dominio de donde se puede obtener una jerarquía de metas. Durante el recorrido por los casos de uso genéricos se seleccionan aquellos que cumplan las condiciones de inclusión y satisfacción de metas según se expone en la sección 3.1.

Para instanciar los parámetros se tienen dos alternativas, se obtienen los datos del modelo del problema o se le solicitan al reutilizador. La determinación automática de estos datos aparece como una búsqueda de patrones. La solicitud de los datos al desarrollador proporciona una vía de realimentación al repositorio. En ambas alternativas, la instanciación de actividades genéricas apoya la ingeniería de requisitos lo cual forma parte de la estrategia de reutilización de requisitos.

4 Conclusiones

En este artículo hemos presentado una aproximación para gestionar y reutilizar assets de requisitos en un repositorio de mecanos. El trabajo continúa nuestra línea de incorporar la reutilización sistemática del software desde etapas tempranas del ciclo de vida. En este caso particular, hemos hecho un avance práctico en materia de reutilización del software a partir de los requisitos. Se ha tomado como base las ideas de Reubenstein [20] y los trabajos de Sutcliffe y Maiden [23] que han propuesto el empleo de modelos genéricos para evitar la construcción de modelos específicos en cada área de aplicación. Estos trabajos se reflejan en nuestro modelo de requisitos genéricos.

La introducción de la dimensión de genericidad mediante modelos genéricos de requisitos ofrece un esquema para gestionar las instancias de assets de dominios particulares en la fase de desarrollo para reutilización. Esta gestión permite recuperar los assets aplicables para un problema dado en la fase de desarrollo con reutilización. Como aspecto resaltante, los modelos genéricos ofrecen la posibilidad de la reutilización inter-dominios con lo cual se explotan mejor los servicios de biblioteca de reutilización. Y además, los modelos genéricos se pueden ofrecer como apoyo a la ingeniería de requisitos para asistir en el desarrollo de especificaciones de requisitos.

La estructura de los assets de requisitos propuesta se ha centrado en los requisitos de funcionalidad con lo que se han dejado de lado a los requisitos no funcionales. No obstante, son muchos los trabajos que señalan la bondad de los escenarios y los casos de uso para permitir su integración con los requisitos no funcionales [3][9][10][13].

El proceso de reutilización de requisitos propuesto garantiza que, ante un problema planteado para abordarse con reutilización, se pueda ofrecer al reutilizador algún modelo para ser instanciado. Esta situación permite el abordaje de problemas complejos mediante composición soluciones con reutilización. No obstante, debemos extender nuestra aproximación con un subproceso para composición de aplicaciones con reutilización del software a partir de los requisitos funcionales, y otro para instanciación de modelos genéricos. Estos subprocesos son parte de nuestro trabajo inmediato.

5 Agradecimientos

Este documento se ha generado con el aporte constructivo del Grupo GIRO, Departamento de Informática, Universidad de Valladolid, y ha sido parcialmente financiado por CICYT(TIC2000-1673-c06-05) como parte del proyecto DOLMEN. Oscar López también agradece a Agencia Española de Cooperación Internacional (AECI) y Ministerio de Ciencia y Tecnología de Costa Rica (MICYT).

6 Referencias

[1] Biggerstaff, T.J. An assessment and analysis of software reuse. Advances in Computers, (34). Academic Press, 1992 Edited by Marshall C. Yovits

[2] Bubenko, J.; Kirikova, M.; "Worlds" in Requirements Acquisition and Modelling. In *4th European-Japanese Seminar on Information Modelling and Knowledge Bases*, Stockholm, 1994.

[3] Cockburn, A. Writing Effective Use Cases. Humans and Tech. Addison-Wesley, Q3 2000. 1999.

[4] Creps, R.; Simos, M.; Prieto-Díaz, R. The STARS Conceptual Framework for Reuse Processes. Defense Advanced Research Projects Agency. November 1992.

[5] Cybulski, J. L.; Patterns in Software Requirements Reuse. Department of Information Systems, University of Melbourne. http://www.dis.unimelb.edu.au/staff/jacob. 1998.

[6] Dardenne, A.; Fickas, S.; Lamsweerde, V.; Goal-directed concept acquisition in requirements elicitation. CIS-TR-91-08, Depart. computer and information science, Univ. Oregon, Mar- 1991.

[7] García, F.J. et alt.; Mecanos: Exposición de resultados y líneas abiertas en la reutilización sistemática del software. Actas Jornadas Iberoamericanas Ingeniería de Requisitos y Ambientes de Software. 1999. pp 193-204.

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

[9] Jacobson, Griss, Jonsson; Software Reuse: Architecture, Process, Organization for Business Success.ACM,1997

[10] Jacobson, I et alt.; Object-Oriented Software Engineering: A Use-Case Driven Approach. Addison-Wesley 1992

[11] Jarke, M.; et alt.; Theories Underlying Requirement Engineering: An Overview of NATURE at Genesis. In Proc. of the IEEE International Symposium on Requirements Engineering, 1993.

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

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

[14] Lamsweerde, A.; Requirements Engineering in the Year 00: A Research Perspective. In Proc. 22nd International Conference on Software Engineering, Limerich, June 2000, ACM Press.

[15] López, O; Laguna, M.A.; Marqués, J.M.; Generación Automática de Casos de Uso para Desarrollo de Software con Reutilización. Actas V Jornadas Ingeniería de Software, Valladolid, 2000. ISBN 84-8448-065-8. pp 89-101.

[16] López, O; Laguna, M.A.; Marqués, J.M.; Reutilización del Software a partir de Requisitos Funcionales en el Modelo Mecano: Comparación de Escenarios. Actas Jornadas Iberoamericanas Ingeniería Requisitos y Ambientes Software, San José, 2001. ISBN9968-32-000. pp 104-116.

[17] Maiden,N.;Sutcliffe,A.;Exploting reusable specification through analogy. In Commun.ACM. 35(4):55-64. 1993

[18] Prieto-Díaz, R.; Classification of Reusable Modules. In Software Reusab., Vol 1. Concepts and Models. Ed. by T. Biggerstaff and Alan J. Perlis. ACM Press, Frontier Series. 1989. pp 99-123.

[19] Rada, R. Reengineering Software, How to Reuse Programming to Build New, State-of-art Software. 2º Edition. USA AMACON. 1999.

[20] Reubenstein, H., Waters R.; The Requirements Apprentice: Automated Assistance for Requirements Acquisition. In IEEE Trans.on Softw Eng., 17(3), pp.226-240, (March 1991). 204

[21] Silva, M., Las Redes de Petri en la Informática y la Automática. Editorial AC. Madrid, 1985.

[22] Sutcliffe, A.; Maiden, N.; Manuel, D.; Supporting Scenario-Based Requirements Engineering In IEEE Transactions on Software Engineering. (24):12. December,1998.

[23] Sutcliffe, A.; Maiden, N.; The Domain Theory for Requirements Engineering. In IEEE Transactions on Software Engineering. (24):3. March,1998.