Informe Final del Proyecto de Investigación
Hacia una Estrategia para Reutilización de Requisitos: El Modelo ReCOTS

Oscar López
ITCR, Sede San Carlos
Ingeniería en Computación
olopez@itcr.ac.cr

Gaudy Esquivel
ITCR, Sede San Carlos,
Ingeniería en Computación
gesquivel@itcr.ac.cr

Miguel A. Laguna
Universidad de Valladolid,
Departamento de Informática
mlaguna@infor.uva.es

Resumen

Para que la reutilización de requisitos contribuya al mejor aprovechamiento de los recursos del desarrollo de software y en la disminución de errores es necesario atacar tres problemas. Primero, la diversidad de técnicas de modelado. Segundo, el bajo nivel de formalidad de los modelos de requisitos. Y, tercero, la carencia de una propuesta validada en la práctica. Por ello, en este proyecto se propuso una investigación tendiente a consolidar una estrategia de reutilización de diagramas semiformales de requisitos. La consolidación de dicha estrategia incluye tres acciones de investigación. Una de estas acciones, la mejora del Modelo ReCOTS (***Re****quirements-based* ***C****ommercial* ***o****ff-****t****he-****S****helf*), es el motivo del presente informe de proyecto, el que se realiza en coordinación con el Departamento de Informática de la Universidad de Valladolid.

- NOMBRE DEL PROYECTO: Hacia una Estrategia para Reutilización de Requisitos: El Modelo ReCOTS
- DEPARTAMENTO RESPONSABLE: Ingeniería en Computación
- PROGRAMA: Centro de Investigaciones en Computación
- INVESTIGADOR RESPONSABLE: Oscar López Villegas
- PERÍODO: Enero de 2004 - Diciembre de 2004

1. Introducción

1.1. Antecedentes

Ante el panorama actual del desarrollo de software, en el que las necesidades de los usuarios empujan al desarrollo de software por la corriente de la complejidad, se realizó el trabajo doctoral publicado en [Lóp03]. En este trabajo se desarrolló un marco para integrar varias técnicas que facilitan la gestión de los requisitos en el contexto de la reutilización sistemática del software. Desde esta perspectiva, se exploró la investigación en el campo de la Ingeniería del Software, que se relaciona principalmente con las áreas de Reutilización del Software, Ingeniería de Requisitos y, en particular, con la intersección de estas áreas que se podría denominar reutilización de requisitos (ver Figura 1). Una serie de conceptos, como el modelado de dominios y del negocio, patrones de requisitos, *frameworks*, *workflows*, e Ingeniería Inversa, recibieron especial atención dentro de este trabajo. Estos conceptos permitieron relacionar coherentemente las áreas de investigación indicadas y fueron puntos de apoyo para desarrollar una *propuesta de reutilización de requisitos.*

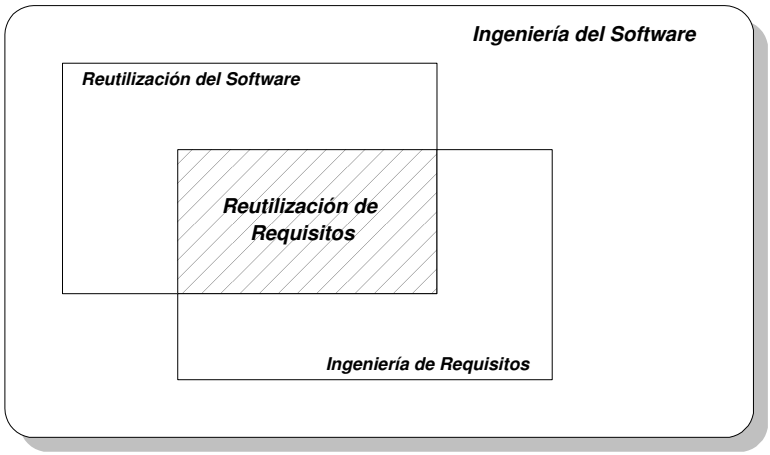

Figura 1: Intersección entre Reutilización del Software e Ingeniería de Requisitos.

El trabajo realizado en [Lóp03] establece la necesidad de *adquirir una mayor experiencia en la reutilización abordada desde los modelos de requisitos.* Por ello, se han propuesto tres líneas de acción para ampliar la experiencia en la reutilización abordada desde los modelos de requisitos. Esas acciones se realizan en estrecha coordinación con el Grupo GIRO de la Universidad de Valladolid[1], que en la actualidad se centra en la organización de software reutilizable siguiendo el enfoque de líneas de productos. Específicamente, en el presente proyecto se pre-

[1]La reutilización en el nivel de los requisitos es una sub-línea del Programa de Investigación GIRO, Universidad de Valladolid, España, para los próximos años (ver el sitio www.giro.infor.uva.es).

tendió la ampliación del modelo de elemento reutilizable, denominado ReCOTS, en la forma que se detalla en la Sección 3.

1.2. Definición del problema

El problema fundamental del desarrollo de software se puede describir como la búsqueda de la *satisfacción real del usuario mediante las prestaciones de los sistemas software*. Impulsado por el intento de satisfacer las demandas del usuario, el proceso de desarrollo del software desencadena una espiral de necesidades. La disciplina económica enseña que las necesidades evolucionan en forma de espiral pues al satisfacerse unas se engendran otras nuevas. En consecuencia, los sistemas software son cada vez más complejos. La creciente complejidad del software entraña el riesgo de que la espiral de necesidades alcance dimensiones catastróficas. Se impone entonces una necesidad mayor, el *contar con un proceso para solventar racionalmente las necesidades del usuario a través del software*.

La necesidad de un proceso racional para el desarrollo de software ha dado origen a la *Ingeniería del Software*. La creciente complejidad del software y la desmesurada demanda de software de calidad ha dado origen a diferentes divisiones dentro de esta disciplina. Para incorporar el aspecto de racionalidad en la industria del software, ha sido relevante el surgimiento de las actividades tanto de *Reutilización Sistemática del Software* como de *Ingeniería de Requisitos*. La reutilización persigue el aprovechamiento de esfuerzos exitosos para desarrollar nuevos productos software. La Ingeniería de Requisitos se encarga de gestionar el proceso de definición de las necesidades del usuario.

1.3. Objetivos

El propósito del proyecto ha sido continuar el trabajo en reutilización de requisitos para contribuir al desarrollo de sistemas software dentro de las cotas de calidad, tiempo y requisitos. Se toma como base una de las opciones de más éxito de Reutilización del Software, que es la aproximación de líneas de producto[2] [CN02]. El desarrollo de líneas de productos se basa en el análisis del dominio para establecer los rasgos comunes y las diferencias entre los miembros de una familia de productos. No obstante, las empresas (y organizaciones) generalmente han aplicado métodos de Ingeniería de Requisitos que intentan desarrollar una especificación para un sistema en particular.

El resultado de la Ingeniería de Requisitos orientado a sistemas (aplicaciones) software individuales en un dominio es un conjunto de especificaciones. Cada una

[2] Además de la aproximación de líneas de producto, la Reutilización del Software se ha abordado con éxito mediante aproximaciones basadas en generación.

de esas especificaciones corresponde a una aplicación software en particular. Y precisamente, la observación general de que existe cierta similitud entre los sistemas que se construyen en un dominio ha potenciado el enfoque de líneas de productos. Lógicamente, dentro del proceso de desarrollo de software, se puede inferir que el desarrollo de soluciones similares es la consecuencia de similitudes en el conjunto de especificaciones de requisitos. De esta manera, el desafío que se plantea consiste en *organizar la información de los requisitos de aplicaciones existentes* para potenciar la estrategia de análisis de dominios. Esta información organizada debe permitir la obtención de especificaciones correctas de nuevas aplicaciones mediante reutilización de *assets* de requisitos.

La organización de la información de los requisitos, con fines de reutilización, es un proceso complejo en el que confluyen tanto aspectos tecnológicos como culturales. La línea de trabajo, o programa de investigación, en reutilización de requisitos se puede desglosar en los siguientes objetivos general y específicos:

Objetivo General: Incorporar la reutilización de requisitos en el proceso de Ingeniería de Requisitos para el incremento de la calidad y la productividad en el desarrollo de especificaciones.

Objetivos Específicos: En términos específicos, la línea de investigación pretende

1. Integrar diversas técnicas de modelado de requisitos en la creación de *assets* dentro de un dominio.
2. Determinar un esquema de descripción rigurosa de requisitos en un dominio.
3. Interpretar relaciones entre requisitos en un conjunto de aplicaciones de un dominio para establecer la forma de organizar los *assets* como una familia de requisitos.
4. Determinar un proceso para la obtención de especificaciones de productos mediante la reutilización de *assets* de requisitos.
5. Desarrollar las herramientas adecuadas para el soporte operativo de las labores de reutilización.

El presente proyecto se relacionó con los objetivos específicos 1, 3 y 5.

2. Revisión de Literatura

El desarrollo del software, como el de cualquier producto, se puede afrontar mediante dos estrategias básicas: (a) *producción artesanal* y (b) *producción industrial*. La producción artesanal se basa en la aplicación de las habilidades humanas para transformar las materias primas y así satisfacer las necesidades de los clientes. La producción industrial se basa en la satisfacción de las necesidades mediante el

ensamblado de una solución a partir de piezas pre-construidas que responden a estándares definidos. La estrategia artesanal permite una mejor satisfacción de las necesidades específicas de cada cliente a cambio de tener que soportar el elevado coste e ineficiencia en el proceso de desarrollo y mantenimiento. La estrategia industrial ofrece menor coste de desarrollo y proceso más eficiente, pero a cambio de un elevado coste de diseño de piezas estándares [Nei92].

La Ingeniería del Software, que consiste en aplicar un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software [IEE99], pretende evolucionar el proceso del ciclo de vida desde la estrategia artesanal hacia la estrategia industrial. Sin embargo, el software presenta características peculiares al ser un producto lógico o simbólico, que no se manufactura, ni se deteriora, más bien se desarrolla y se adapta a los cambios del medio [Rad99, Pre02]. Estas particularidades determinan la necesidad de diferentes aproximaciones «ingenieriles» para el desarrollo del software. Una de las aproximaciones más prometedoras es la *Reutilización del Software*.

¿Qué es reutilización de software?: La reutilización del software es una estrategia industrial, opuesta a la artesanal, para abordar el desarrollo de aplicaciones con base en una biblioteca de componentes y un proceso eficiente para reducir los costes de producción [Nei92]. El concepto de componente designa a un elemento reutilizable que realiza una funcionalidad determinada y que se comunica a través de interfaces bien definidas [BW98, DW99]. Probablemente la reutilización se origina de dos propuestas pioneras presentadas a finales de los 60's. En 1968, Bemer propuso en la empresa General Electric la creación de una *factoría de software* con el afán de mejorar la productividad de los programadores [IAM97]. También en 1968, McIlroy [McI76] de la empresa AT&T propuso la *producción masiva de rutinas* de código que fueran catalogadas en bibliotecas de reutilización, para su posterior empleo en el ensamblado industrial del software. Ambas propuestas presentan dos características relevantes:

- Enfatizan en la necesidad de una estrategia industrial para abordar la producción del software.
- Se fundamentan en la sinonimia entre componentes hardware y componentes software.

La necesidad de abordar industrialmente la producción del software es un hecho aceptado con amplitud ante la presión para disponer de sistemas software de alta calidad. Sin embargo, la analogía entre componentes hardware y componentes software no es aceptable [Nei92]. Aunque la reutilización obedece a un principio sencillo, como lo es el aprovechar los esfuerzos previos y exitosos, se requiere un enfoque particular, propio y adaptado a la naturaleza del software, para el aprovechamiento efectivo de los elementos

reutilizables en el proceso de producción de nuevo software. Este enfoque debe permitir la *selección, especialización e integración* de productos software que hayan sido intencionalmente *diseñados, desarrollados y documentados* para servir como materia prima para el desarrollo de nuevos productos software [Kru92, Gar00].

Reutilización de requisitos: La gama de trabajos que abordan la reutilización de software es amplia e incluye distintos niveles de abstracción: diseño, esquemas de bases de datos, modelos de dominio, requisitos, gestión, economía, cultura, leyes [Kar95]. El proceso de *Ingeniería de la Reutilización* se estructura mediante dos procesos complementarios: *Ingeniería del Dominio* e *Ingeniería de la Aplicación*, según la propuesta del Departamento de Defensa de USA [CSPD92].

La reutilización de requisitos se ha planteado en [Lóp03] como una alternativa para la Ingeniería del Dominio que se basa en un *Modelo de Requisitos Reutilizables*, un *Modelo de Proceso* y un conjunto de *herramientas de soporte*. El Modelo de Requisitos Reutilizables se denomina ReCOTS (***Re****quirements-based* ***C****ommercial* ***o****ff-****t****he-****S****helf*). El Modelo ReCOTS se caracteriza por admitir en su estructura los requisitos que se representan mediante seis técnicas de modelado ampliamente conocidas (escenarios, casos de uso, diagramas de actividades, diagramas de flujos de datos, diagramas documentos-tareas y diagramas de flujos de trabajo o *workflows*). El Modelo de Proceso para reutilización de requisitos se denomina DIRECTORY (***Di****agram-based* ***Re****usable* ***C****omponents* ***To****wards* ***R****equirements* Qualit***Y***). El proceso DIRECTORY es una guía para el aprovechamiento de los diagramas de requisitos existentes en un dominio. El soporte para reutilizar requisitos es proporcionado por una herramienta CASE, a la que se denomina el Entorno R^2.

3. Metodología

Actualmente la reutilización del software se realiza en la forma de líneas de productos. Esta estrategia de reutilización permite incrementar la productividad, reducir costes de desarrollo de cada producto, obtener mejores estimaciones referentes al proceso de desarrollo y mejorar la calidad de los productos [Bos02, CN02]. El desarrollo de líneas de productos involucra el establecimiento de una *familia de requisitos* que refleja los resultados de un análisis del dominio para identificar *comunalidades*, *variabilidades* y dependencias entre los requisitos. Diferentes aproximaciones de líneas de productos, como FODA [KCH$^+$90], PuLSE [BFK$^+$99] y ODM [SCK$^+$96], se fundamentan en un análisis previo del dominio. Sin embargo, en muchos dominios se ha recopilado información de requisitos mediante métodos de Ingeniería de Requisitos, que se enfocan generalmente hacia un producto

en particular y no desde una perspectiva institucional de línea de productos. Para abordar la reutilización sistemática de esa información generada en el proceso de Ingeniería de Requisitos se requiere el soporte para organizar familias de requisitos. Para ello se ha considerado necesario incidir mediante investigación en *tres acciones*:

1. Ampliación del Modelo ReCOTS para incluir otras técnicas de modelado de requisitos y productos que se derivan de estos.

2. Construcción de dominios para obtener conclusiones válidas respecto a la efectividad de la estrategia de reutilización de diagramas de requisitos.

3. Mejora del proceso DIRECTORY en los aspectos de verificación, cualificación y análisis léxico.

Para expresar estas acciones paralelas se utiliza la notación SADT [Ros77], ver Figura 2. Los rectángulos representan actividades. Las flechas que llegan desde la izquierda representan entradas. Las flechas que llegan desde arriba indican mecanismos de control. Las que llegan desde abajo representan recursos. Y las que salen hacia la derecha representan salidas o productos de la actividad.

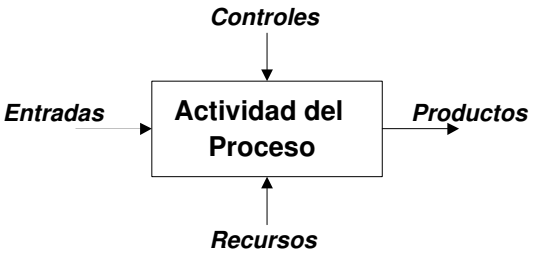

Figura 2: Representación de actividades según SADT.

El marco general[3] que se propuso para investigación en reutilización de requisitos se describe en la Figura 3. Como su nombre indica, el primer subproceso se encarga de preparar los requisitos para ser reutilizados, así como su reutilización misma. Por ello, las entradas iniciales al primer subproceso son Especificaciones Existentes, Conocimiento del Dominio, la Propuesta DIRECTORY y Necesidades del Cliente. Estas entradas sirven de base para generar salidas que en su conjunto constituyen un modelo del dominio. En el segundo subproceso se toma como entradas la Propuesta ReCOTS, Especificaciones Existentes y el Entorno R^2 para obtener como salida el Modelo ReCOTS (que sirve de control en los subprocesos 1 y 3), así como Metaclases de Productos Derivados y el Entorno R^2 mejorado. El tercer subproceso toma el Conocimiento de Ingeniería de Dominios [SCK+96] y las Propuestas de DIRECTORY [Lóp03], Métricas (de requisitos) [HS02], LEL

[3]Se refiere a la línea o programa de investigación donde se incluye la labor de investigación que se aborda en el presente proyecto.

(Léxico Extendido del Lenguaje) [LHDK00] y Patrones (regularidades independientes del dominio) [RDL01] para obtener una serie de modelos que sirven de controles para el primer subproceso.

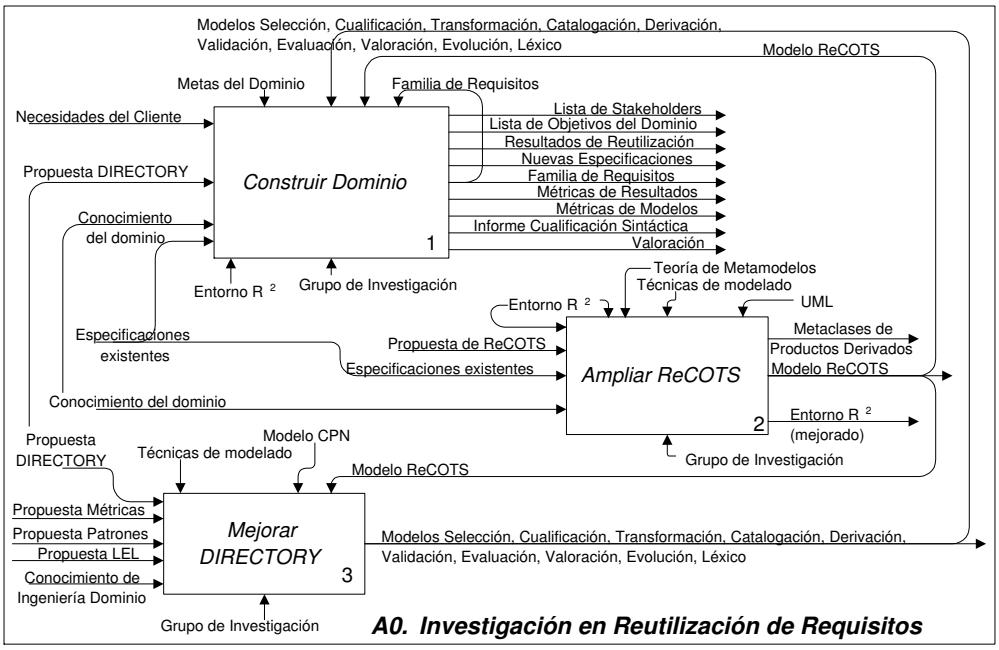

Figura 3: Marco general para investigación en Reutilización de Requisitos.

Todos los sub-procesos de la Figura 3 toman como recurso el Grupo de Investigación, el que está distribuido en Valladolid y Costa Rica (y abierto a la colaboración e interacción con otros grupos afines). Dentro de ese Grupo de Investigación se debe contar con los siguientes roles: (a) Ingeniero de Requisitos, (b) Ingeniero de Aplicación, (c) Experto del Dominio y (d) Cliente.

La labor fundamental para reutilización sistemática de software es la de Construir Dominios. No es viable técnicamente la reutilización sistemática sino se cuenta con elementos reutilizables debidamente representados, identificados, cualificados, clasificados y almacenados en un repositorio. Sin embargo, para la labor de Construir Dominios es imprescindible contar con:

- Un modelo de elemento reutilizable, que en el presente caso se denomina el Modelo ReCOTS.
- Un modelo de proceso para aprovechar los elementos reutilizables, lo que se ha denominado el Proceso DIRECTORY.

Tanto el Modelo ReCOTS, como el Proceso DIRECTORY, deben ser ampliados y mejorados para emprender la labor de Construir Dominios. En el presente proyecto se abordó la labor de ampliar el Modelo ReCOTS.

3.1. Ampliación del Modelo ReCOTS

El Modelo ReCOTS está esquematizado como un metamodelo de requisitos del software centrado en la perspectiva de los Requisitos-C [Rom90]. La finalidad del Modelo ReCOTS es proporcionar un esquema de descripción de elevado nivel de abstracción para los Diagramas de Requisitos que se pueden organizar y almacenar como *assets* en un repositorio. El metamodelado dentro del Modelo ReCOTS constituye el fundamento (conceptual) para atacar el problema de la reutilización de requisitos.

El Modelo ReCOTS permite describir diagramas de requisitos ampliamente conocidos, como Diagramas de Workflow, Casos de Uso, Escenarios, dDT, DFD y Diagramas de Actividades. La descripción que se logra con el metamodelado permite identificar categorías de unidades de modelado, y sus relaciones, dentro de los diagramas estudiados. Además, el metamodelo propuesto soporta la descripción del carácter semántico y estructural de las relaciones entre los elementos de modelado de requisitos. De esta manera, la perspectiva integradora de los requisitos ofrecida permite describir a los diagramas de requisitos como conjuntos de conceptos y conjuntos de relaciones. Las instancias de la metaclase Tarea aparecen como un elemento común y esencial en los diferentes modelos de requisitos. La metaclase Tarea aporta una perspectiva de integración que se evidencia en la descripción estandarizada que aporta una Plantilla de Especificación de Secuencia.

El trabajo del metamodelo constituye un hito para el establecimiento de una aproximación de reutilización sistemática de los requisitos. La descripción de alto nivel de abstracción y la perspectiva integradora del Modelo ReCOTS son la base del proceso de reutilización de requisitos. El metamodelo es además el esquema conceptual del repositorio de requisitos. Sin embargo, el metamodelo debe ser ampliado para describir con mayor amplitud el modelo del dominio y según se propone en la Figura 4.

Para ampliar el Modelo ReCOTS se realizaron las etapas siguientes:

Etapa 1. Describir Productos Derivados: Esta etapa consistió en tomar como entradas el Conocimiento del Dominio, las Especificaciones Existentes y la Propuesta de ReCOTS para producir como salida las Metaclases de Productos Derivados. Estas metaclases son descripciones de productos de diseño de las aplicaciones. Se incorporaron los Diagramas de Clases, Diagramas de Interacción y Diagramas de Secuencia de UML [OMG01] como productos de diseño. Esta etapa tomó como control el Modelo ReCOTS.

Etapa 2. Refinar Relaciones de Variabilidad: Tomando como controles la Teoría de Metamodelos, las Técnicas de Modelado y UML, y a partir de la Propuesta ReCOTS y las Metaclases de Productos Derivados, se amplió el Modelo ReCOTS incorporando más formas de modelar la variabilidad en los requisitos. De este modo, el Modelo ReCOTS permite modelar características

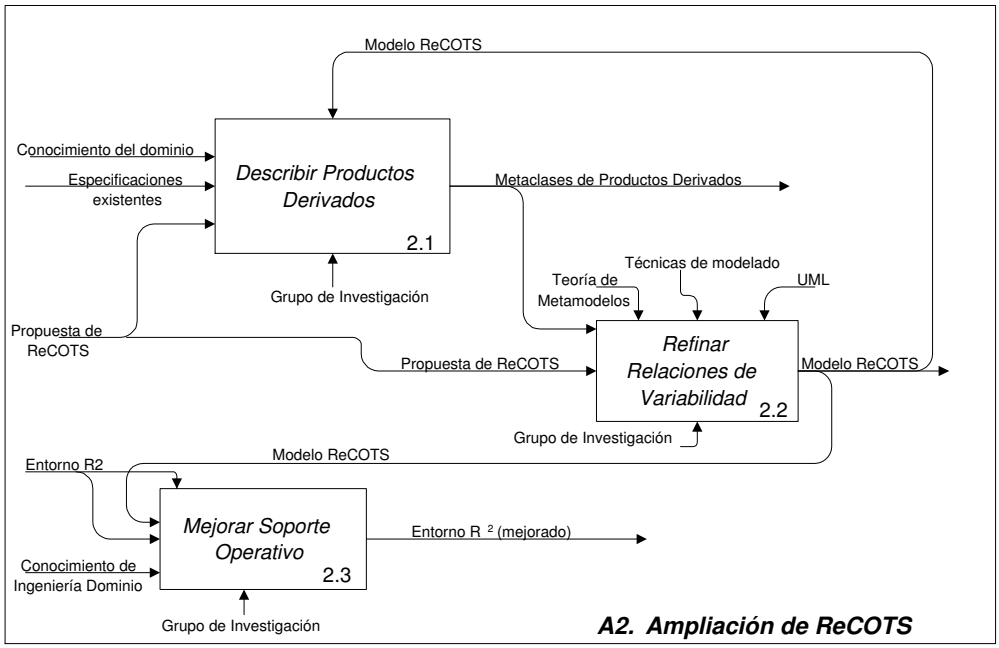

Figura 4: Proceso de ampliación del modelo ReCOTS.

opcionales, optativas, obligatorias y múltiples de los sistemas, y además se refinó esa variabilidad para incluir no sólo la selectividad de características sino también las *habilidades y preferencias* de los usuarios.

Etapa 3. Mejorar Soporte Operativo: El Conocimiento del Dominio y el Entorno R^2 sirvieron de insumos para obtener un mejor soporte operativo para el proceso de reutilización de requisitos. Esta etapa produjo como salida el Entorno R^2 mejorado. En la Figura 5 se muestra la interfaz de usuario del Entorno R^2.

En el Cuadro 1 se presenta el desglose y distribución de las etapas en el tiempo.

4. Resultados y Discusión

La labor total aquí presentada de reutilización de requisitos se aborda como un programa de investigación formado por tres diferentes proyectos o etapas, según se mostró en la Figura 3. Una de esas etapas ha sido la base para el proyecto que aquí se reseña. Es decir, el presente proyecto está limitado al ser una parte dentro de un programa de investigación. Sin embargo, es la intención de los firmantes que este proyecto sea el inicio para profundizar en la reutilización de requisitos. Este inicio deberá continuar mediante futuros proyectos individuales, hasta completar

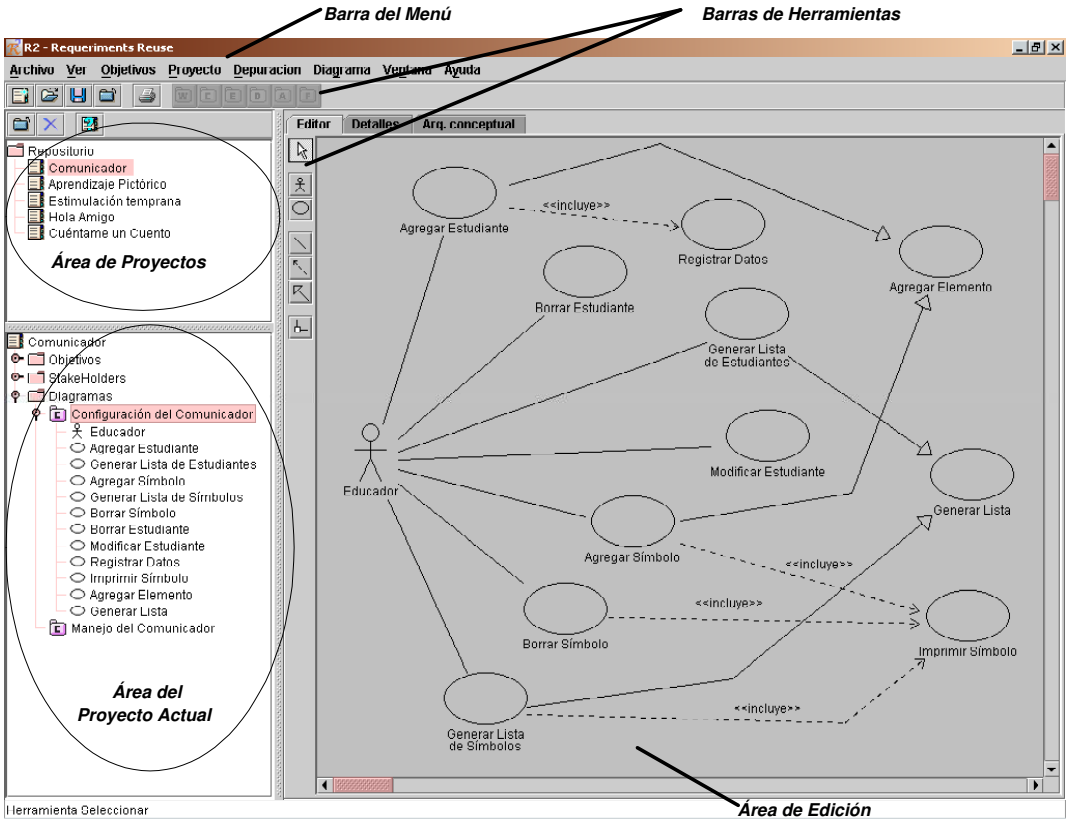

Figura 5: Interfaz de usuario del entorno de reutilización de requisitos.

las labores presentadas en la Figura 3.

4.1. Descripción de los Productos Derivados

Los Productos Derivados son aquellos artefactos derivables de los diagramas de requisitos y que representan un refinamiento o disminución del nivel de abstracción de estos. Los Productos Derivados son artefactos que se pueden concebir más en la fase de diseño que en la de requisitos dentro del proceso de software. La descripción de Productos Derivados ha llevado a la ampliación del esquema conceptual del modelo de Representación de Requisitos (el metamodelo de requisitos, conocido como el Modelo ReCOTS). La ampliación permite integrar en una descripción tanto productos de análisis como de diseño, y de esta forma incorporar Diagramas de Clases, Diagramas de Colaboración y Diagramas de Secuencia como productos reutilizables.

ReCOTS es un modelo de estructura de reutilización, y está formado por Modelos de Representación de Requisitos, Productos Derivados de estos, y Objetivos del Dominio. Los Modelos de Representación son una estructura que describe req-

ETAPAS	MESES DEL AÑO 2004					
	1-2	3-4	5-6	7-8	9-10	11-12
1. Describir Productos Derivados						
1.1 Diagrama de secuencia	√					
1.2 Diagrama de colaboración	√					
1.3 Diagrama de Clases		√				
2. Refinar variabilidad						
2.1 Incluir habilidades			√			
2.2 Incluir preferencias				√		
3. Mejorar soporte operativo						
3.1 Organización y recuperación			√	√		
3.2 Nuevos Diagramas	√	√	√			
3.3 Modelado de variabilidad				√	√	
Redacción del Informe Final						√

Tabla 1: Desglose y distribución de las etapas en el tiempo.

uisitos funcionales y no funcionales desde la perspectiva de los clientes. Los Productos Derivados, según ya se ha dicho, son cualquier producto de software que conceptualmente representa un refinamiento de un conjunto de Modelos de Representación. Y los Objetivos del Dominio son una intención general aceptada como válida dentro de un dominio y que debe ser satisfecha mediante el desarrollo de aplicaciones software para lo cual debe ser desglosada (el objetivo del dominio) en otros objetivos o modelado mediante un diagrama de requisitos.

El concepto de Objetivo del Dominio concuerda con la definición de *requisito* dada por A. Davis[4]: «una característica externamente observable de un sistema deseado». Y todo ello coincide con lo que Sommerville [Som01] denomina "requisito de usuario" (*user requirement*). Y desde una perspectiva general, que clasifica a las metas u objetivos (*goals*) en cuatro categorías: cognitivas (por ejemplo, estar de acuerdo con algo o alguien), físicas (por ejemplo, retornar o realizar algo), "dirigidas al sistema" (*system-driven*) (por ejemplo, el sistema almacena cierta información), y comunicativas (por ejemplo, solicitar algo), los Objetivos del Dominio se clasifican como *system-driven*.

En la Figura 6 se presenta el modelo básico de ReCOTS. Además de los conceptos de Modelo de Representación, Producto Derivado y Objetivo de Dominio, la Figura muestra las metaclases *Productor* e *Identificador de Sistema*. Productor representa a la organización y/o al autor(es) responsable(s) de la creación, evaluación y mantenimiento de las estructuras reutilizables. La metaclase Identificador de Sistema representa la definición lógica de un proyecto software dentro del do-

[4]Alan M. Davis and Ann M. Hickey. How to Select the "Right" Requirements Elicitation Technique. Half-day Tutorial. VII Jornadas de Ingeniería del Software y Bases de Datos (JISBD-2002). El Escorial, 19 de noviembre de 2002.

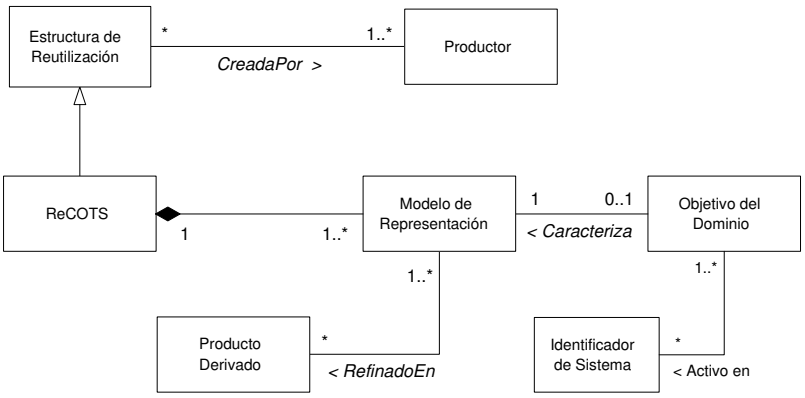

Figura 6: Modelo básico de ReCOTS, expresado en UML.

minio en el que se reutilizan requisitos.

La clase Estructura de Reutilización está relacionada con la clase Productor mediante la asociación *CreadaPor*, de forma que toda estructura reutilizable tenga asociado al menos un responsable, ya sea una organización o una persona concreta.

Un ReCOTS se modela como una composición dado que cada uno de los componentes de un ReCOTS no puede ser parte de otros ReCOTS. Además, la eliminación de un ReCOTS como estructura reutilizable conlleva obligatoriamente a la eliminación de sus componentes.

La metaclase Modelo de Representación está relacionada con la metaclase Producto Derivado mediante la (meta)asociación *RefinadoEn*, de forma que todo Producto Derivado tenga asociado al menos un Modelo de Representación (de requisitos).

4.2. Los Modelos de Representación

Un ReCOTS está formado por Modelos de Representación (de requisitos), Productos Derivados, y Objetivos del Dominio. La (meta)entidad Modelo de Representación es el centro del esquema conceptual para describir y gestionar los requisitos reutilizables que se representan mediante diversas técnicas de modelado.

Las técnicas de modelado muestran perspectivas diferentes de los requisitos. Para integrar esas perspectivas dentro de una estrategia de reutilización de requisitos, se debe determinar el contenido de los Modelos de Representación. En la Figura 7 se representan las (meta)entidades asociadas a los Modelos de Representación (de requisitos), que se utiliza para describir los diferentes diagramas de requisitos. Estas (meta)entidades son las siguientes:

- La *Unidad de Modelado*, utilizada para describir las unidades contenidas en los diagramas de requisitos,
- La *Relación Unidad* que permite enlazar las Unidades de Modelado unas con otras,

- La *Relación UnidadModelo* que permite describir Unidades de Modelado complejas, y
- La *Relación Modelo* que se utiliza para enlazar Objetivos del Dominio.

Las metaclases *Relación Unidad, Relación Unidad Modelo* y *Relación Modelo* representan diferentes relaciones estructurales que describen aspectos semánticos dentro de la propuesta de reutilización de requisitos. Estas relaciones son descritas a través de tres tipos de relaciones estructurales:

Relación Unidad: Caracteriza los enlaces semánticos entre las Unidades de Modelado de requisitos dentro de los Modelos de Representación

Relación Unidad Modelo: Describe las relaciones entre Unidades de Modelado y Modelos de Representación.

Relación Modelo: Describe los enlaces entre los elementos de la estructura de conocimiento del dominio, que a su vez son la puerta de entrada a los Modelos de Representación

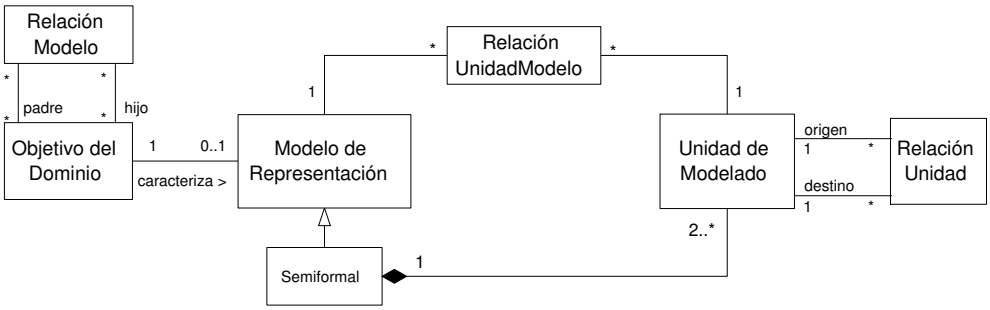

Figura 7: Elementos relacionados con los Modelos de Representación de requisitos.

Un Modelo de Representación está compuesto por elementos de menor granularidad que se denominan Unidades de Modelado. La metaclase Modelo de Representación representa la entidad Modelo de Representación. Un Modelo de Representación se modela como una composición porque cada uno de los componentes de un Modelo de Representación no puede ser parte de otros Modelos de Representación, además la eliminación de un Modelo de Representación conlleva obligatoriamente a la eliminación de sus componentes.

4.3. Las Unidades de Modelado

Los modelos de requisitos que representan el *Universo del Discurso* están formados por Unidades de Modelado. Las Unidades de Modelado incluidas en los Modelos de Representación (de requisitos) considerados en este proyecto se clasifican en siete categorías (ver Figura 8):

Actividad: Representa una labor o proceso que se realiza dentro del dominio. Se distinguen dos tipos de actividades:

- Tarea: Es una unidad formada por la agregación de otras actividades.
- Acción: Representa una actividad atómica, es decir, una actividad que no presenta subdivisiones.

Sujeto: Representa a los actores que intervienen en el sistema, que pueden ser personas, unidades organizacionales o sistemas autónomos.

Objetivo: Representa las intenciones específicas de usuarios y sistema, en el contexto de la interacción entre ellos.

Limitación: Información que restringe la funcionalidad del sistema. Puede ser de tipo temporal o de disponibilidad de algún recurso, que a su vez puede ser recursos físicos o datos.

Estado: Representa una situación dinámica (un espacio o período de tiempo) durante el cual se satisface alguna condición, se realiza alguna labor, o se espera por un evento.

Conector: Representa un criterio de ordenamiento semántico entre Unidades de Modelado y puede ser de tres tipos:

- Lineal: Indica un ordenamiento secuencial entre dos Unidades de Modelado, es decir, una a partir de la otra.
- Unión: Representa la confluencia de un número $N > 0$ de flujos para producir un flujo de llegada a una Unidad de Modelado. La Unión puede ser de dos tipos:
  - Unión Total: Indica que se requiere que el control llegue a los N flujos de entrada para generar un flujo de salida.
  - Unión Parcial: Indica que el control debe llegar a P flujos, $P < N$, para generar un flujo de salida.
- Bifurcación: Indica la generación de $N > 0$ flujos de salida a partir de un flujo de entrada. La Bifurcación puede ser de dos tipos:
  - Bifurcación Total: Indica que cuando el control llega al flujo de entrada se produce los N flujos de salida.
  - Bifurcación Parcial: Indica que al llegar el control al flujo de entrada se generan P flujos de salida, $P < N$. Un caso particular de dos salidas en una bifurcación parcial refleja una situación de exclusión mutua, lo que permite modelar situaciones de opcionalidad como en los diagramas de flujo.

Unidad Derivada: Representa un elemento software que refina a un conjunto de Unidades de Modelado y puede ser de diferentes tipos, dependiendo del paradigma de programación a utilizar. A la fecha se ha estudiado el *Paradigma de Orientación al Objeto*. Una Unidad Derivada que corresponde a ese Paradigma es la categoría (metaclase) Abstracción Objetual, que representa un concepto que encapsula comunicación, proceso e información[5] según se detalla a continuación:

- Comunicación: Representa el punto de contacto para la interacción con otros elementos del sistema.
- Proceso: Son los métodos u operaciones que determinan cómo debe actuar o responder el elemento software. Los procesos pueden ser de tres tipos:
  - Síncrono (el proceso permanece desactivo hasta que reciba respuesta).
  - Plano (el proceso activo no espera respuesta).
  - Asíncrono (el proceso no espera respuesta, pero se mantiene activo).
- Información: Son las propiedades o datos que encapsula un objeto. Estos datos determinan el estado del objeto.

Los Productos Derivados son un refinamiento de uno o más Modelos de Representación, y a su vez el Producto Derivado está formado por Unidades de Modelado (las mismas utilizadas para describir las unidades contenidas en los diagramas de requisitos, sólo que a nivel de diagramas de diseño se obtiene un menor nivel de abstracción). Por tanto, la ampliación del metamodelo ha consistido en una revisión de sus metaclases, lo que ha llevado fundamentalmente en la incorporación de la metaclase Unidad Derivada, la que permite representar productos de diseño, ver Figura 8.

4.4. El Modelo ReCOTS ampliado

El Modelo ReCOTS ampliado recoge en un único modelo los elementos que conforman la estructura de un Componente de Requisitos Reutilizables, el que permite incorporar elementos de diseño. Este modelo constituye el esquema conceptual de modelo de repositorio, donde aparece inmerso el ReCOTS, como modelo de componente reutilizable, y la información necesaria para su selección, recuperación, clasificación, cualificación, y mantenimiento.

En la Figura 9 se muestran todas las entidades que conforman el Modelo ReCOTS ampliado (o lo que es lo mismo, el modelo de repositorio para almacenar requisitos reutilizables, incorporando artefactos de diseño), expresado mediante un

[5] Además debe cumplir tres características básicas: estar basado en objetos y clases y ser capaz de mantener relaciones de herencia de clases.

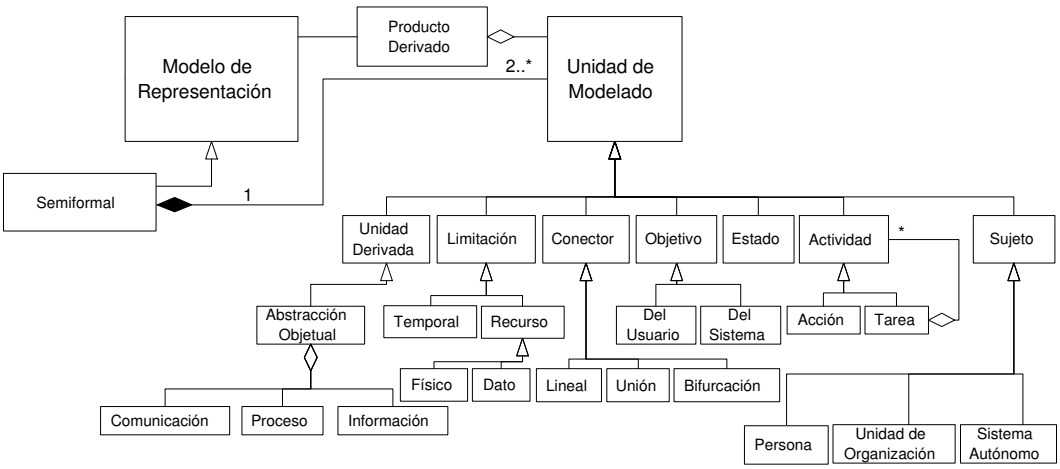

Figura 8: Tipos de Unidades de Modelado que se incluyen en los Modelos de Representación y en los Productos Derivados.

diagrama de clases de UML. ReCOTS se ubica en el Nivel Meta y es instanciado (en el Nivel Modelo) como Familia de Requisitos.

4.5. Categorías de Modelos de Representación de Requisitos

El Modelo ReCOTS ampliado constituye el esquema conceptual del repositorio de requisitos y productos derivados reutilizables. La misión de este modelo es servir de esquema de integración y gestión de requisitos y productos derivados, que son modelados mediante diferentes técnicas de modelado. En el Nivel Meta se ubican las metaclases Modelo de Representación y Producto Derivado (que según se ha visto, forman parte del Modelo ReCOTS). En el Nivel Modelo se expresan los modelos que proponen distintas técnicas de modelado de requisitos y diseño, éstos son descritos mediante las metaclases. Esas técnicas de modelado se instancian en el Nivel Datos representando diagramas concretos en el contexto del desarrollo de aplicaciones software específicas, ver Figura 10.

En el contexto de este proyecto se abordan modelos de tipo Semiformal. Específicamente se tratan los modelos del subtipo Modelo de Comportamiento, los cuales se instancian como diagramas que representan la interacción entre el sistema y entidades externas (por ejemplo, los Diagramas de Escenarios [BL01] y Casos de Uso [OMG01]), diagramas que representan los requisitos haciendo énfasis en las funciones de transformación de datos (como por ejemplo los Diagramas Documentos-Tareas [CHL87] y Flujos de Datos [DeM79]) y diagramas que permiten reflejar la gestión de recursos de la organización (por ejemplo, los Diagramas de Workflow [WfM99] y de Actividades [OMG01]). Además, en la ampliación del Modelo ReCOTS, se incorporan los Diagramas de Clases, Diagramas de Secuencia y Diagramas de Colaboración [OMG01].

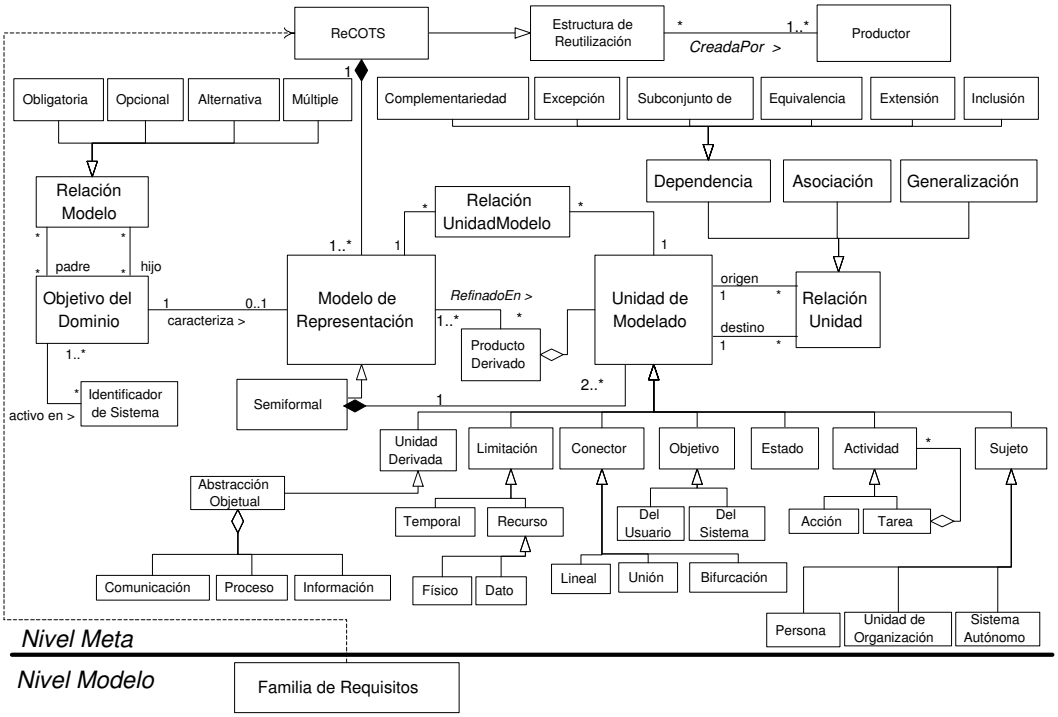

Figura 9: Modelo ReCOTS ampliado (expresado en UML).

4.6. Instanciación del Diagrama de Casos de Uso

El Diagrama de Casos de Uso [OMG01] se representa como un paquete que instancia la metaclase Modelo de Comportamiento, ver Figura 11. Dentro de este paquete se instancia la metaclase Tarea como Caso de Uso. El elemento Caso de Uso se relaciona con un Evento Disparador, una Precondición, una Postcondición, un Resultado, una Meta y un Actor, que son instancias de los diferentes subtipos de la metaclase Unidad de Modelado. Estos subtipos están enlazados al Caso de Uso mediante relaciones de Asociación que representan instancias de un tipo de la clase Relación Unidad. Además, un Caso de Uso muestra relaciones de Dependencia con otro Caso de Uso (Extensión e Inclusión) y de Generalización.

De acuerdo con la especificación del lenguaje UML 1.x [OMG01], un Caso de Uso tiene el propósito de describir un servicio que proporciona una entidad, con independencia de los detalles de su estructura interna. El servicio se describe como una secuencia completa que es iniciada por un actor. Es decir, cuando la secuencia se ejecuta totalmente la entidad queda en disposición de iniciar nuevamente la misma secuencia. El Caso de Uso comprende posibles variantes de la secuencia, por ejemplo secuencias alternas, comportamiento excepcional, manejo de errores, etc. Por todo esto, la descripción de un Caso de Uso generalmente se hace con el apoyo de estructuras de lenguaje natural restringido como ha sido propuesto en [Coc00, Dur00] y en absoluta congruencia con lo recomendado en [OMG01].

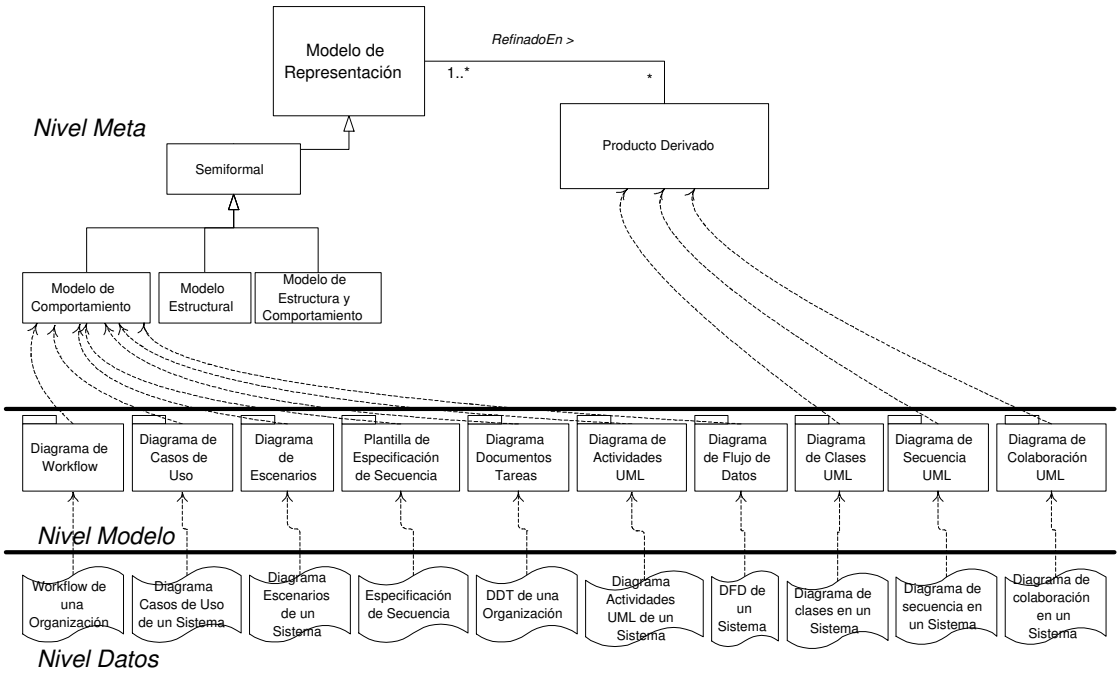

Figura 10: Modelos de Representación de requisitos y Productos Derivados.

Concordante con estos planteamientos, en la Figura 11 se incluye el paquete Plantilla de Especificación de Secuencia [Dur00] como una instancia de Representación de Requisitos. Esta Plantilla permite especificar las realizaciones de satisfacción y de excepción de cada Caso de Uso (es decir, de las instancias de la clase Caso de Uso dentro del paquete Diagrama de Casos de Uso) mediante una instancia de Relación Unidad Modelo.

4.7. Instanciación del Diagrama de Clases

Un Diagrama de Clases describe los tipos de objetos que hay en el sistema y las diversas clases de relaciones estáticas que existen entre ellos. El Diagrama de Clases se representa como un paquete que instancia la metaclase Producto Derivado, ver Figura 12. Este paquete contiene el elemento Clase como instancia de la metaclase Abstracción Objetual. A su vez una Clase tiene Interfaz, Restricciones, Operaciones y Atributos, estos elementos son instancias de las metaclases Comunicación, Limitación, Proceso e Información respectivamente. Como se puede observar en el paquete de Diagrama de Clases, el elemento Clase tiene asociado una clase Tipo, que representa las variantes de los tipos de clases existentes; estas no se instancian, ya que se representan con la instanciación del elemento Clase.

Las clases están enlazadas mediante relaciones de Asociación, Generalización y Dependencia; estas son instancias de la metaclase Relación Unidad, que a su vez tiene información asociada sobre Rol (nombres para los roles que desempeñan los

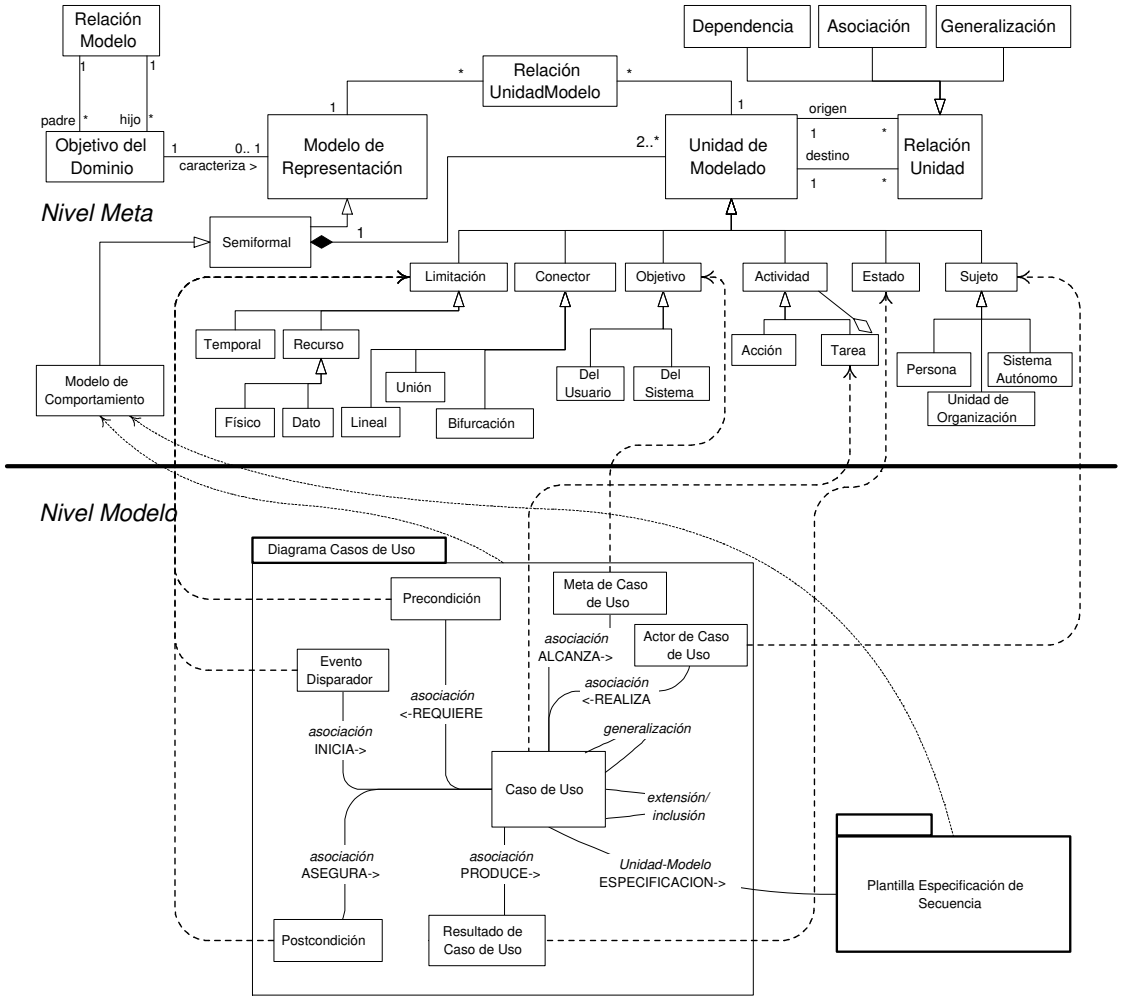

Figura 11: Instanciación del Diagrama de Casos de Uso.

objetos en la asociación), Multiplicidad (Indica la cantidad de objetos que participarán en la relación dada) y Navegabilidad (posibilidad de navegar unidireccionalmente en una asociación) existente entre Clases.

4.8. Instanciación del Diagrama de Secuencia

Un Diagrama de Secuencia muestra los objetos y actores que participan en una colaboración encima de las líneas de punto. La línea representa el tiempo visto por el objeto: es la línea de vida del objeto. El Diagrama de Secuencia se representa como un paquete que instancia la metaclase Producto Derivado, ver Figura 13. Este paquete tiene los elementos Actor y Objeto que son instancias de la metaclase Abstracción Objetual; ambos elementos envían y reciben Mensajes. La clase mensaje es una instancia de un Proceso y tiene asociado los elementos Valor de

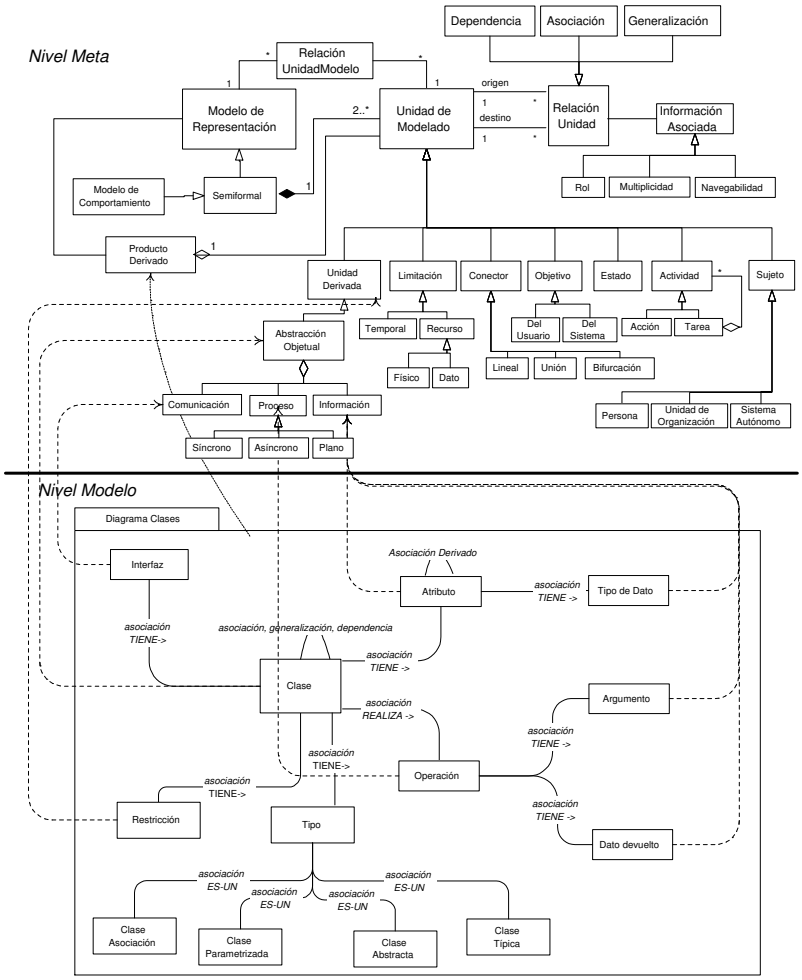

Figura 12: Instanciación del Diagrama de Clases.

Retorno, Argumento, Condición, Indicador de Secuencia, Iteración y Restricción de Tiempo. Estos elementos asociados al mensaje son instanciados en las meta-clases Información para la clase Valor de Retorno y Argumento, Limitación en la clase Condición, Conector Lineal para la clase Indicador de Secuencia, Estado para la clase instancia de Iteración y Limitación Temporal para la clase de Restricción de Tiempo. Los Objetos del paquete Diagrama de Secuencia tiene Activaciones y Línea de Vida las cuales son instancias de la clase Estado.

La relación entre los Objetos en el paquete Diagrama de Colaboración es de asociación, que es una instancia de un tipo de la Clase Relación Unidad.

4.9. Instanciación del Diagrama de Colaboración

Un Diagrama de Colaboración consiste en todos los objetos que interactúan para ejecutar alguna tarea, junto con los enlaces entre ellos. El Diagrama de Colab-

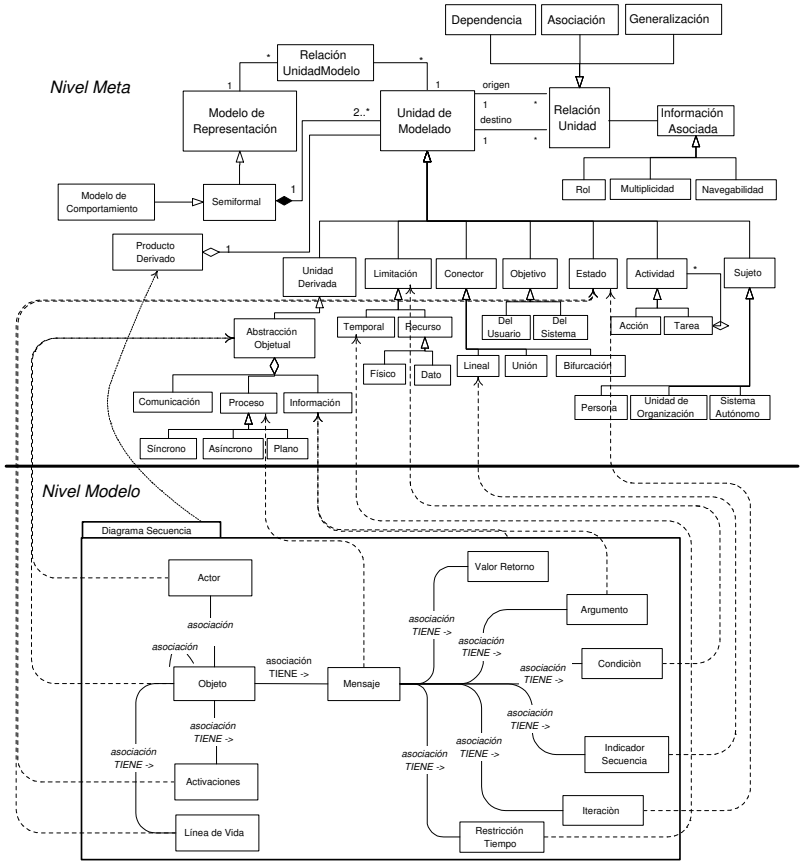

Figura 13: Instanciación del Diagrama de Secuencia.

oración se representa como un paquete que instancia la metaclase Producto Derivado, ver Figura 14. Los elementos Actor y Objeto del paquete El paquete Diagrama de Colaboración tiene dos elementos: que son instancias de la metaclase Abstracción Objetual. Estos elementos envían y reciben Mensajes que tienen características de Secuencia e Iteración, y contienen Argumentos y Valores de retorno. El elemento Mensaje es una instancia de la metaclase Proceso y los elementos Valor de Retorno y Argumento son instancias de la metaclase Información; Indicador de Secuencia e Iteración respectivamente son instancias de Conector Lineal y Estado.

La relación entre los Objetos en el paquete Diagrama de Colaboración es de Asociación, la cual es una instancia de un tipo de la clase Relación Unidad.

4.10. Relaciones de Variabilidad

Un Producto Derivado puede ser cualquier producto que se obtiene a partir de un Modelo de Representación. Por ejemplo, de un conjunto de casos de uso se pueden derivar diagramas de clases. La derivación puede ser manual o automática. En cualquier caso, se debe indicar explícitamente la relación de refinamiento entre

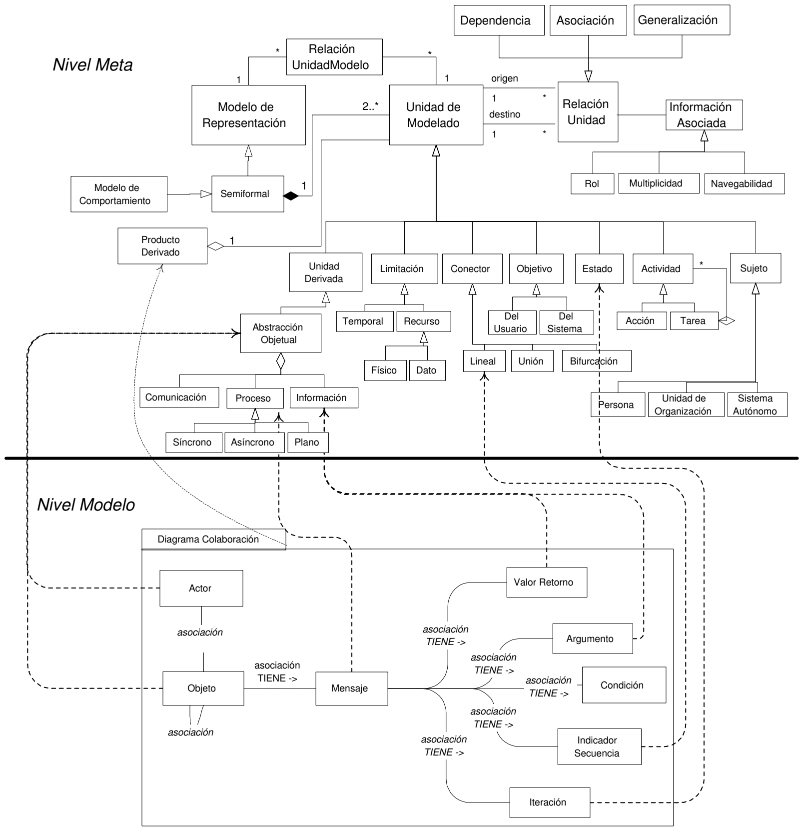

Figura 14: Instanciación del Diagrama de Colaboración.

Productos Derivados y Modelos de Representación.

Dado que el Modelo ReCOTS permite modelar características opcionales, optativas, obligatorias y múltiples de los sistemas, y ante la intención de modelar no sólo la selectividad de características sino también las *habilidades y preferencias* de los usuarios, se decidió modelar estas últimas mediante las mismas relaciones de ReCOTS. Es decir, las habilidades y preferencias y preferencias se expresan mediante diferentes Productos Derivados, de modo que la relación entre Modelos de Representación y Productos Derivados expresa las habilidades y preferencias del usuario.

4.11. Mejora del Soporte Operativo

Dada la complejidad de las labores de reutilización del software (véase por ejemplo [Mar98]) se requiere un robusto soporte operativo, lo que aquí se realiza mediante el Entorno R^2. Por ello, ha sido muy importante en el éxito del presente proyecto de investigación el haber contado con el apoyo de dos (2) estudiantes de Práctica de Especialidad de Ingeniería en Computación, quienes laboraron en mejorar el soporte operativo, según se presentó en la Tabla 1. El entorno, en su

estado actual, permite la edición de todos los diagramas indicados en la Figura 10 y la conformación de una familia de requisitos. Además, permite recorrer el grafo de requisitos y reutilizar los elementos según la selección del reutilizador.

En resumen, las principales mejoras que se han hecho al Entorno R^2 son:

- Al crear un nuevo proyecto se muestra una vista completa o resumida del árbol de objetivos creado para el dominio, el usuario debe seleccionar el o los objetivos a reutilizar en su nuevo proyecto.
- La creación de un diagrama de requisitos está asociada a un único objetivo de dominio.
- Permite la creación y edición de los diagramas de la categoría de productos derivados. Estos deben enlazarse únicamente de los diagramas de requisitos.
- Permite la creación y edición de diagramas de clases, secuencia y colaboración.

5. Conclusiones y recomendaciones

El tema central de este proyecto es la reutilización de requisitos, que puede aplicarse en cualquier dominio que cumpla con las características de ser restringido y bien entendido, de acuerdo con Krueger [Kru92], y que posee la suficiente importancia para justificar el esfuerzo de reutilización.

Para abordar la diversidad de técnicas de modelado de requisitos, y la existencia de diferentes niveles de descripción de los mismos, el Modelo ReCOTS (ampliado en este proyecto) posee un ámbito de acción restringido en los siguientes términos:

- Admite en su estructura los requisitos que se representan mediante seis técnicas de modelado ampliamente conocidas (escenarios, casos de uso, diagramas de actividades, diagramas de flujos de datos, diagramas documentos-tareas y diagramas de flujos de trabajo o *workflows*).
- Se basa en el metamodelado como esquema conceptual que permite:
  - La descripción e integración de los diagramas mediante un modelo común, y
  - La organización de los requisitos dentro de una familia de diagramas.
- Incorpora artefactos o productos software que conceptualmente representan un refinamiento, de los diagramas de requisitos. A estos refinamientos se les denomina Productos Derivados.

El Modelo ReCOTS está esquematizado como un metamodelo de requisitos del software centrado en la perspectiva de los Requisitos-C [Rom90]. Su finalidad es proporcionar un esquema de descripción de elevado nivel de abstracción para

los Diagramas de Requisitos que se pueden organizar y almacenar como *assets* en un repositorio. La labor desarrollada se enmarca dentro del establecimiento de una *propuesta de reutilización de requisitos*.

El Modelo ReCOTS permite describir Diagramas de Workflow, Casos de Uso, Escenarios, dDT, DFD y Diagramas de Actividades. Además, con la ampliación obtenida, se pueden describir Diagramas de Clases, Diagramas de Secuencia y Diagramas de Colaboración. La descripción que se logra con el metamodelado permite identificar categorías de unidades de modelado, y sus relaciones, dentro de los diagramas estudiados. Además, el metamodelo propuesto soporta la descripción del carácter semántico y estructural de las relaciones entre los elementos de modelado de requisitos. De esta manera, la perspectiva integradora ofrecida permite describir a los diagramas de requisitos y diseño como conjuntos de conceptos y conjuntos de relaciones.

El trabajo realizado en este proyecto constituye un avance importante en el camino de establecer una aproximación de reutilización sistemática de los requisitos. La descripción de alto nivel de abstracción y la perspectiva integradora del Modelo ReCOTS (requisitos y diseño) son la base para el proceso de reutilización de requisitos. El metamodelo es además el esquema conceptual del repositorio de requisitos que se implementa dentro del entorno R^2.

6. Aportes y alcances

Aportes: La reutilización de requisitos se puede considerar como una alternativa para la Ingeniería del Dominio. Al estar ubicada en el nivel de abstracción conceptual, la reutilización de requisitos persigue el apoyo a las labores de Ingeniería de Requisitos, las que son críticas para el desarrollo del software pues *consumen recursos y son propensas a errores* con consecuencias nefastas en el resto del ciclo de vida. Para avanzar en el establecimiento de una estrategia que aprovecha los diagramas de requisitos existentes en un dominio, es necesario abordar la construcción de dominios, así como mejorar las técnicas disponibles. Por ello, en este documento se han propuesto tres acciones de investigación.

Las tres acciones planteadas son: (1) Construcción de un dominio, (2) Ampliación del modelo de elemento reutilizable (el Modelo ReCOTS), y (3) Mejora de técnicas específicas para reutilizar requisitos. La construcción de dominios es una labor estratégica para validar las técnicas de reutilización. La ampliación del Modelo ReCOTS es importante dado que la descripción de los elementos reutilizables constituye la base para aprovechar las diversas representaciones de requisitos en el desarrollo de especificaciones con reutilización. La mejora de las técnicas específicas constituye la clave para avanzar en el proceso de reutilización de requisitos.

El aporte de este proyecto se pueden resumir en tres:

1. Descripción de elevado nivel de abstracción para productos de diseño de software, como diagramas de interacción y de clases.
2. Mejora de un modelo de elemento reutilizable, como lo es el Modelo ReCOTS.
3. Mejora de una herramienta para el soporte operativo de la reutilización, como lo es el Entorno R^2.

Alcances: En las últimas décadas se ha venido reconociendo de manera amplia que los requisitos del software deben ser gestionados adecuadamente para el éxito de los proyectos de desarrollo de software en general [vL00], y de reutilización del software en particular [Cyb98]. Numerosas investigaciones han mostrado la importancia de los requisitos del software con respecto a la calidad [BT76], disminución de fallos [SGI98], y coste de corrección de errores [KS98]. De acuerdo con Brooks [Bro87], la etapa más difícil del software, y la que más negativamente puede afectar las etapas posteriores, es precisamente la que consiste en definir qué software desarrollar. Esta aseveración refleja la necesidad del tratamiento objetivo de los requisitos y ha despertado el interés de la comunidad de Ingeniería del Software, cuya respuesta ha sido el fomentar la creación de la disciplina de Ingeniería de Requisitos [Dur00]. Con la reutilización de requisitos en dominios se puede mejorar la productividad del desarrollo de software. Si los ingenieros de requisitos pueden reutilizar requisitos para familias de productos, se pueden desarrollar las especificaciones y el software en menor tiempo, a la vez que se reduce la probabilidad de errores en los requisitos.

7. Aspectos técnicos y administrativos

7.1. Cumplimiento de objetivos

Se confirma la coincidencia entre los objetivos planteados en la propuesta inicial del proyecto y los objetivos que fueron alcanzados durante la ejecución del mismo.

7.2. Limitaciones y problemas encontrados

No han habido problemas que tuvieran un efecto significativo en la ejecución del proyecto. La formulación de la metodología ha sido adecuada, lo mismo que la estimación de los tiempos designados a las distintas actividades. No han existido problemas con el equipo humano a cargo del proyecto o actividad, a excepción de la retirada del investigador Apolinar González, quien se retiró de la Institución. No se dieron limitaciones en la disponibilidad de la infraestructura y equipo destinado al proyecto, ni dificultades en trámites administrativos. Tampoco han existido problemas de coordinación con entes internos o externos involucrados en el proyecto.

7.3. Observaciones generales y recomendaciones

Este proyecto constituye una continuación del proyecto doctoral del investigador principal. Esta situación significa que el proyecto no ha partido de cero, sino que ya existía un camino recorrido, el que ha sido continuado. Esta situación también significa que el proyecto se ha realizado en contacto estrecho con el Departamento de Informática de la Universidad de Valladolid, lo que representa un importante vínculo de colaboración internacional.

8. BIBLIOGRAFÍA

Referencias

[BFK+99] J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J. M. DeBaud. PuLSE: A methodology to develop software product lines. In *Proceedings of the Fifth ACM SIGSOFT Symposium on Software Reusability (SSR'99)*, pages 122–131, Los Angeles, CA, USA, May 1999. ACM.

[BL01] K. Breitman and J. Leite. Requirements elicitation through scenarios: A hands on tutorial. Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro, 2001.

[Bos02] J. Bosch. *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. ACM Press. Addison-Wesley, May 2002.

[Bro87] F. Brooks. No silver bullet: Essence and accident of software engineering. *Computer*, 20(4):10–19, 1987.

[BT76] T. E. Bell and T. A. Thayer. Software requirements: Are they really a problem? In *Proceedings: 2nd International Conference on Software Engineering*, pages 61–68. IEEE Computer Society Press, 1976.

[BW98] Alan W. Brown and Kurt C. Wallnau. The current state of CBSE. *IEEE Software*, 15(5):37–46, September/October 1998.

[CHL87] A. Collongues, J. Hugues, and B. Laroche. *Merise. Méthode de conception*. Dunod, France, 1987.

[CN02] P. Clements and L. M. Northrop. *Software Product Lines: Practices and Patterns*. The SEI series in software engineering. Addison-Wesley, 2002.

[Coc00] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley, Boston, 2000.

[CSPD92] R. Creps, M. Simos, and R. Prieto-Díaz. The STARS conceptual framework for reuse processes. In *Proceedings of STARS'92*, November 1992.

[Cyb98] J. L. Cybulski. Patterns in software requirements reuse. Technical report, Department of Information Systems. University of Melbourne, July 1998.

[DeM79] T. DeMarco. *Structured Analysis and System Specification*. Prentice-Hall publishers-Yourdon, Inc, 1979. Also published in/as: Yourdon, Inc., New York, 1978.

[Dur00] A. Durán. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información*. PhD thesis, Universidad de Sevilla, Spain, 2000.

[DW99] Desmond Francis D'Souza and Alan Cameron Wills. *Objects, Components and Frameworks with UML. The Catalysis Approach*. Object Technology Series. Addison Wesley, 1999.

[Gar00] F. J. García. *Modelo de Reutilización Soportado por Estructuras Complejas de Reutilización Denominadas Mecanos*. PhD thesis, Universidad de Salamanca, Spain, 2000.

[HS02] B. Henderson-Sellers. Requirements engineering: The object-oriented context. Keynote Address. V Workshop on Requirements Engineering (WER-2002) Valencia, Spain, November 2002.

[IAM97] P. Bottcher I. Aaen and L. Mathiassen. The software factory: Contributions and illusions. downloaded on May 21th, 2002, from http://iris.informatik.gu.se/conference/iris20/39.htm, 1997.

[IEE99] IEEE. *IEEE Software Engineering Standard Collection. 1999 Edition*. IEEE Computer Society Press, 1999.

[Kar95] E. Karlsson, editor. *Software Reuse. A Holistic Approach*. Wiley Series in Software Based Systems. John Wiley and Sons Ltd, 1995.

[KCH+90] K. Kang, S. Cohen, J. Hess, W. E. Novak, and A. Peterson. Feature-Oriented Domain Analysis (FODA). Feasibility study. Technical Report CMU/SEI-90-TR21 (ESD-90-TR-222), Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, November 1990.

[Kru92] C. W. Krueger. Software reuse. *ACM Computing Surveys*, 24(2):131–183, June 1992.

[KS98] G. Kotonya and I. Sommerville. *Requeriments Engineering: Processes and Techniques*. USA Wiley, 1998.

[LHDK00] J. Leite, G. Hadad, J. Doorn, and G. Kaplan. A scenario construction process. *Requirements Engineering. Springer-Verlag London Limited*, 5(2000):38–61, 2000.

[Lóp03] O. López. *Reutilización de Requisitos para Incremento de la Calidad y Poductividad en el Desarrollo de Especificaciones*. PhD thesis, Universidad de Valladolid, Spain, 2003.

[Mar98] José Manuel Marqués. Soporte operativo para la reutilización del software: Repositorios y clasificación. In José Luis Barros and Antonio Domínguez, editors, *Actas Del Curso Ingeniería de Software Y Reutilización: Aspectos Dinámicos Y Generación Automática*, Escuela Universitaria de Ingeniería Técnica en Informática de Gestión. Edificio Politécnico - Ourense, Julio 1998. Univeridade De Vigo. Ourense, del 6 al 10 de Julio de 1998.

[McI76] M. D. McIlroy. Mass-produced software components. In J. M. Buxton, P.Ñaur, and B. Randell, editors, *Software Engineering Concepts and Techniques; 1968 NATO Conference on Software Engineering*, pages 88–98. Van Nostrand Reinhold, 1976.

[Nei92] J. M. Neighbors. The evolution from software components to domain analysis. *International Journal of Software Engineering and Knowledge Engineering*, 2(3):325–354, October 1992.

[OMG01] OMG. Unified modeling language specification. Version 1.4. Technical report, Object Management Group Inc., September 2001. http://cgi.omg.org/docs/formal/01-09-67.pdf.

[Pre02] R. Pressman. *Ingeniería del software: un enfoque práctico*. McGraw-Hill, 5th edition, Febrero 2002.

[Rad99] R. Rada. *Reengineering Software, How to Reuse Programming to Build New, State-of-art Software*. USA AMACON, 2nd edition, 1999.

[RDL01] M. Ridao, J. Doorn, and J. Leite. Domain independent regularities in scenarios. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, pages 120–127. IEEE Computer Society, August 2001.

[Rom90] H. D. Rombach. Sofware specifications: A framework. SEI Curriculum Module. Technical Report SEI-CM-11-2.1, Software Engineering Institute, Carnegie Mellon University, January 1990.

[Ros77] D. T. Ross. Structured analysis (SA): A language for communicating ideas. *IEEE Transactions on Software Engineering*, 3(1):16–34, January 1977. Special collection on Requirement Analysis.

[SCK+96] M. Simos, D. Creps, C. Klingler, L. Levine, and D. Allemang. Organization domain modeling (ODM) guidebook - version 2.0. Technical Report STARS-VC-A025/001/00, Lockheed Martin Tactical Defense Systems, 9255 Wellington Road Manassas, VA 22110-4121, June 1996.

[SGI98] The Standish Group International. CHAOS: a receipe for success. http://www.standishgroup.com/sample_research/chaos1998.pdf, 1998.

[Som01] I. Sommerville. *Software Engineering*. Addison-Wesley, USA, 6th edition, 2001.

[vL00] A. van Lamsweerde. Requirements engineering in the year 00: A research perspective. In *Proceedings of the 22nd. International Conference on Software Engineering*, Limerich, June 2000. ACM Press.

[WfM99] WfMC. The workflow management coalition. Terminology and Glossary. Document Number WFMC-TC-1011. United Kingdom, 65 pages, February 1999.